AFRL-RI-RS-TR-2017-140

FAST COGNITIVE AND TASK ORIENTED, ITERATIVE DATA DISPLAY (FACTOID)

CHARLES STARK DRAPER LABORATORY

JUNE 2017

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2017-140 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
PETER A. JEDRYSIK
Work Unit Manager

/ S /
JULIE BRICHACEK
Chief, Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUNE 2017	FINAL TECHNICAL REPORT	SEP 2012 – MAR 2017

4. TITLE AND SUBTITLE

FAST COGNITIVE AND TASK ORIENTED, ITERATIVE DATA DISPLAY (FACTOID)

5a. CONTRACT NUMBER
FA8750-12-C-0293

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Joshua Poore

5d. PROJECT NUMBER
XDAT

5e. TASK NUMBER
A0

5f. WORK UNIT NUMBER
17

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Charles Stark Draper Laboratory
555 Technology Sq.
Cambridge, MA 02137

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RISB
525 Brooks Road
Rome NY 13441-4505

DARPA/I2O
675 North Randolph St.
Arlington, VA 22203-2114

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2017-140

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

On the DARPA XDATA program, Draper developed new approaches to understanding the usability and adoptability of data analytics applications. Draper developed the Software as a Sensor™ (SensSoft) capability, which turns analytics software applications themselves into measurement mediums for usability and adoptability. These measurements were used to develop and validate models of how users sequence and integrate features of software applications. Draper integrated SensSoft logging into applications developed by 6 other XDATA performers and used the newly developed metrics to provide comparative ranking and analysis to the XDATA program. To promote these advances in the broader community, Draper transitioned this technology to the Apache Software Foundation, and has planned follow-on efforts with DARPA, IARPA, and NGA efforts.

15. SUBJECT TERMS
Usability Testing; Software as a Sensor; Software evaluation; User modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**PETER A. JEDRYSIK**
U	U	U	U	68	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1. SUMMARY

At the start of the XDATA program, Dr. Chris White challenged Draper to innovate a new capability and apparatus for understanding the usability and adoptability of applications designed to support data analytics professionals. While Draper previously demonstrated the utility of ingesting software activity logs into pre-processing methods for psychophysiological research in user engagement, Dr. White further challenged us to accomplish this task without the use of traditional laboratory sensors. Essentially, Draper was challenged to develop methods and apparatus that would turn analytics software applications themselves into measurement mediums for usability and adoptability. After four years of scientific research and software development, Draper has demonstrated this very capability and has engineered an apparatus to provide this capability as a service. We call this capability Software as a Sensor™ (SensSoft).

As one of the world's leaders and innovators in instrumentation, signal processing, and both modeling and simulation, Draper takes very seriously the vision of software that serves as a sensor for gathering data about user behavior, modeling it to uncover both user-specific and canonical behavioral patterns that indicate proficient end-use. Draper developed for the XDATA program:

- An Analytic Logging Engine to facilitate collection of this data from both prototype and mature software environments
- Modeling approaches that capture how users sequence and integrate features of software applications
- Validated metrics for quantifying how well users understand how to use applications to perform real-world tasks
- A testing framework for performing usability evaluation at large scales.

When Dr. Wade Shen assumed control of XDATA program, he challenged Draper to reduce this technology to practice, and provide interfaces into modeling techniques and metrics such that software developers could make use of them to improve their applications. To meet this challenge Draper:

- Engineered its Analytic Logging Engine to require minimal effort for deployment
- Developed software stack to support both analysis, for direct interface with user activity logs, and visual analytics to extract insights that can be used to iteratively improve applications
- Configured a constellation of components into an easy-to-deploy package

Apache Software as a Sensor™ (SensSoft) is now a viable product, available to everyone through the Apache Software Foundation (ASF) that is undergoing transition at the National Geospatial Intelligence Agency (NGA) through a CRADA agreement, and is being used for other Science & Technology (S&T) efforts at DARPA and IARPA.

2. INTRODUCTIONS

Draper's efforts on the XDATA program fall into three main tasks: Software Development; Publicity and Community Development; Evaluation and Human Subject Research Activities. These are described in depth in the sections below.

2.1. Software Development

Draper's software development activities incorporate all work to develop a user activity logging apparatus, and supporting elements: analytics stack; application program interfaces (APIs); visualization and visual analytics; experiment management services. This task also includes documentation and software repository management. Consistent with our XDATA contractual requirements, all software developed under contract is licensed for open-source use. Draper has exceeded this requirement by transitioning all code developed on the XDATA contract to the Apache Software Foundation (ASF), the world's premier open-source software foundation.

2.2. Evaluation and Human Subjects Research Activities

Under direction of Dr. Chris White and Dr. Wade Shen, Draper's role under Technical Area 2 of XDATA was both to innovate new methods and apparatus for understanding usability, as well as to use these new methods in service of evaluating core XDATA program products developed by other performers. This activity incorporates efforts to innovate new methods of evaluating tools and performing evaluations, including: managing an extensible, multi-site human subjects research protocol, approved by a human subject internal review board (HSIRB); coordinating and executing data collection activities, both small and massive; developing experimental artifacts, such as experimental tasks and questionnaires; managing and integrating laboratory equipment; algorithm and analytics development; hypothesis testing; producing evaluation reports.

2.3. Publicity and Community Development

In the fourth year of the XDATA program, Draper worked to foster a community of interest around our open-source software as a transition activity. This activity incorporates efforts to increase the public awareness of Software as a Sensor™, including: work to transition Software as a Sensor™ to other agencies (e.g., NGA); work to transition SensSoft to the ASF; development of both corporate and Apache marketing and documentation websites; generation of marketing materials; social media campaigns; academic/industry conference attendance, demonstrations; and public exhibitions.

3. METHODS, ASSUMPTIONS, AND PROCEDURES

3.1. Software Development

In developing open-source software technology, Draper sought to design and develop applications that are light-weight, easy to maintain, and developed from other open-source components licensed under maximally permissible licenses, so as to reduce conflicts with our own licensing strategy. Below we discuss the classes of software developed, and class-specific assumptions. Descriptions and delineation of features that comprise the specific software products (artifacts) developed for XDATA can be found in Section 4.

3.1.1. User Activity Logging Apparatus. In order to develop an apparatus for capturing user activities through software platforms, Draper understood the following design assumptions:

- The apparatus would need to incorporate sufficient granularity in logs of describing human use that would enable robust signal processing and modeling.
- The apparatus would need to be compatible with a range of different software applications.
- A single embodiment of the apparatus would likely only be able to serve programs written in a specific language.
- The apparatus would need to connect with a database capable of indexing single log messages and managing logs.
- The database serving the apparatus would need to have interface end-points so that log data could be transmitted or utilized by other processes (analytics).
- At least in prototype versions of the apparatus, adaptations to the source code of applications would need be made.

3.1.2. Experimental Management. In order to effectively execute and manage both small and large scale experiments, software would need to be designed to carry out these functions. In developing this software we operated under the following assumptions:

- The software would need to support registration features for participants to enroll in research studies.
- Registration information (participant identifiers) would need be passed to other data collection services (e.g., user activity logs, questionnaires, etc.).
- The software would need to support pathing options for routing participants into different tasks.
- The software would need to “poll” other services, and provide merge operations for various data sets.
- The software would need experimenter interfaces for both tracking, and enrolling participants, as well as for performing analyses and basic pre-processing.

3.1.3. Developer Tools. In year 2, Draper successfully demonstrated that with sufficient granularity in user activity logs, models and metrics could not only describe user behavior, but quantitatively describe both efficient and inefficient user behavior. In year 3, we realized that while useful from an evaluation perspective, the consumer of these models and metrics would largely be program or project managers. Thus, at the end of our third year on the XDATA program, we realized that to actually recommend changes that would *improve* software usability new approaches to modeling and interfacing with user activity logs would be needed to provide insights to developers, so that improvements might be implemented. In the latter part of the third and the whole of the fourth year of the XDATA program, Draper sought to define and implement software applications and new modeling techniques that would support developers and allow them to digest insights gleaned from user activity modeling. We operated under the following assumptions:

- Developers would need to interact with both processed and raw data (user activity logs), using software packages that were commonly used for analytics and querying large databases (e.g., Python).
- Developers would need a way to visualize or produce visual analytics that would help lead them to insights.
- New scalable modeling approaches would need be developed that would better support visualization and visual analytics.

3.2. Evaluation and Human Subjects Research Activities

Throughout the program, Draper was tasked to host, support and execute evaluation activities to support the larger XDATA program. However, given that Draper was a TA2 performer and challenged to develop and reduce to practice *novel* methods and apparatus for performing such evaluations, the role of these evaluations was two-fold: 1) to assess other performers' work products on the basis of usability, and 2) to provide useful data so that novel methods and apparatus could be developed and validated.

3.2.1. Human Subjects Research Protocol. In order to make strong inferences about the utility and performance of novel methods for assessing analytic software usability, Draper would need to collect data from humans under a research activity, necessitating human subjects research protections oversight and approvals by both local and US Government (USG) HSIRB. This was also a contractual requirement, stipulated by DARPA. All protocol documentation, approvals, and continuing review documents were provided as deliverables.

3.2.2. Data Collection and Methods. In order to organize data collection and make meaningful inferences about the differences between different XDATA applications and the validity of novel models and metrics for usability, we devised a scalable experimental design with both between- and within-subjects conditions. This design allowed for compartmentalizing a few critical effects to reduce confounds in findings. First, this design scales with the number of applications (Factor 2), allowing us to test a wide range of XDATA applications,

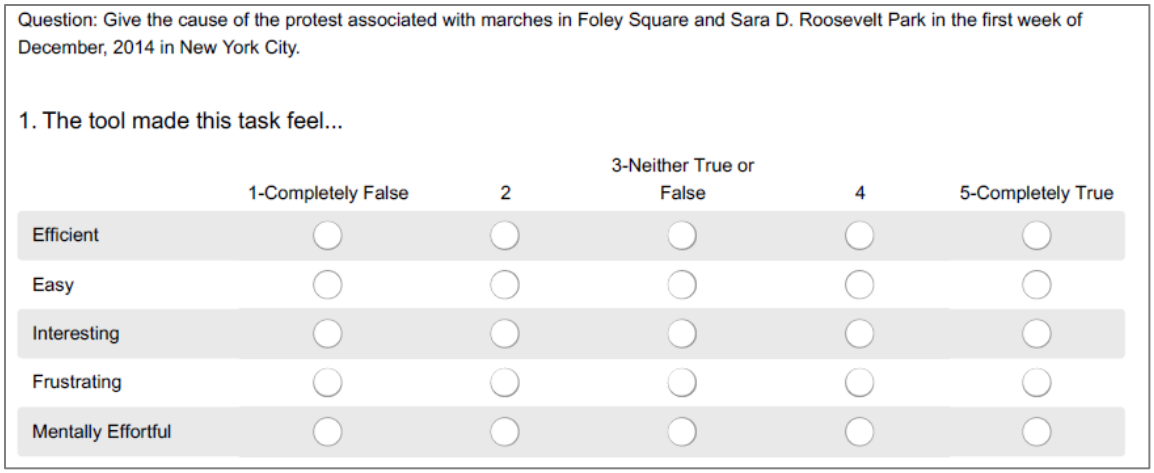
and by incorporating a within-subjects condition (multiple exposures), this provides use with sufficient experimental power to offset scale at the between-subjects level. Second, the within-subjects conditions (Factor 1) allowed us to understand whether specific tasks were more difficult than others, owing to the nascent functionality of the applications. Finally, a counterbalancing factor (Factor 3), allowed us to ensure that the order of tasks given to participants systematically affected their performance in repeated exposures. This experimental design was used to facilitate human subjects testing and data collection throughout the program.

The sequence of tasks distributed to research participants was also common throughout data collection events:

1. *Consent [5mins]:* participants consent to participate or not.

2. *Intake Questionnaire* [30mins max]: composed of a number of surveys soliciting Demographics, expertise, job/analyst experience, personality (pertaining to problem solving and inference)[1-7] and problem solving aptitude [8, 9].

3. *Application Testing with Operational Task (1)* [30mins max]: allowed research participants to interact with the application, with accompanying data within the context of operationally-relevant tasks paired with the data (and challenge problems), *not specific applications*. Operational tasks were composed of a sequence of 5 questions, requiring participants to utilize the application in complex ways to ascertain the answer. Operational tasks were developed largely without the applications under test, with support from other staff (e.g., Qntfy) to provide verifiable tasks that could be accomplished in “state-of-practice” analysis tools (e.g., iPython Notebook). Answers to tasks were known in advance and ground truth answers were verified by both Draper as well as other XDATA personnel (e.g., support staff from Giant Oak and Qntfy). Operational task questions were largely multiple choice response formats (no fewer than 4 response choices). Where possible, we utilized free-response formats, when free-response answers would be easy to parse and easy to adjudicate with little risk of false negatives for correct answers (true positive).


Question: Give the cause of the protest associated with marches in Foley Square and Sara D. Roosevelt Park in the first week of December, 2014 in New York City.

1. The tool made this task feel...

	1-Completely False	2	3-Neither True or False	4	5-Completely True
Efficient	○	○	○	○	○
Easy	○	○	○	○	○
Interesting	○	○	○	○	○
Frustrating	○	○	○	○	○
Mentally Effortful	○	○	○	○	○

Figure 1. Sample Operational Task Component Question

4. *Post-Task Questionnaire (1)* [5-10mins]: completed by participants following each Operational Task. Surveyed participants regarding how much mental/cognitive load the tool introduced to the task, how difficult the task was independent of the tool [10], how engaged they felt with the task [11], and how much they enjoyed the task.

5. *Application Testing with Operational Task (2)* [30mins max]*:* a second Operational Task and opportunity for participants to engage with the application. This task is different, in terms of question content, however, is designed to utilize the same data used for the first task.

6. *Post-Task Questionnaire* (2) [5-10mins]: a second post-task questionnaire, identical to the first, soliciting responses related to the participants' second exposure to the application.

7. *User Comments* [5-10mins]*:* using Nielson's revised design heuristics for subjectively evaluating user-interfaces [12], we collected open-ended responses for each of 10 Heuristics. This provided fast feedback and user comments for immediate dissemination to XDATA Tool Developers.

8. *Debriefing*: an HSIRB-approved script detailing the study hypotheses, purposes in greater depth for participants.

All procedures could be completed within a 1.5-2 hour period of time, including consent procedures. In addition to questionnaire data and free-response data collected from participants, we collected user activity logs through our UserALE service, while participants interacted with applications. In experiments in years 2 and 3 of the XDATA program, Draper also used physiological monitoring protocols as part of data collection, in service of activity log metric development. The goal being to be able to give context to users

subjective reports tied directly to observable behavior during actual application use.

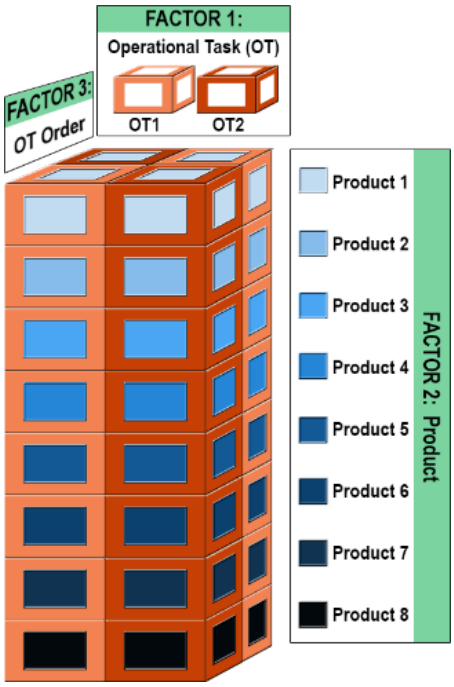


Figure 2. Experimental Design for Human Subjects Data Collection.

Data was collected from human research participants in three separate testing events, during various points in the course of the XDATA program:

1. During the XDATA 2014 Summer Workshop, Draper completed 33 user testing sessions from 33 unique users, with 8 XDATA prototype applications, each using one or more unique datasets. Participants were furnished by XDATA as part of program outreach and transition endeavors. Due to application error, only 16-22 of participants were usable in analyses. Physiological data was also collected from users in this event.

2. During the XDATA 2015 Summer Workshop, Draper completed 38 user testing sessions from 36 unique users, with 6 different applications (Aperture, FEAT, Minerva, Neon, Newman, Resonant). Participants were furnished by XDATA as part of program outreach and transition endeavors. Due to server crashes and participant non-compliance, task data for one or two tasks was sometimes unusable. Draper collected 3 more User Testing sessions following the Summer Workshop in order to fill these gaps so that there were a roughly equal number of participants allocated to each application for analysis. This resulted in a dataset with 33 fully complete cases (performance data, subjective reports, and activity logs), and 36 cases with performance data and activity logs. Comments, however, were not collected from the 3 additional volunteers; all 3 opted out of those measures. Physiological data was also collected from users in this event.

3. At the end of XDATA year 3 (Q4 2015) and throughout XDATA year 4 (2016-2017), Draper conducted a Massive Online Testing (MOT) event of XDATA year 3 applications (Aperture, FEAT, Minerva, Neon, Newman, Resonant). Testing was conducted using applications built by Draper to facilitate (see Findings & Results) online human subjects research at scale. The virtual assets for this data collection event were furnished by DARPA via contractors—these assets include Amazon Web Services (AWS) resources, such as virtual machines, load-balancers, and a web site designed to generate interest in XDATA (participants were routed through this site) (see Figure 3).

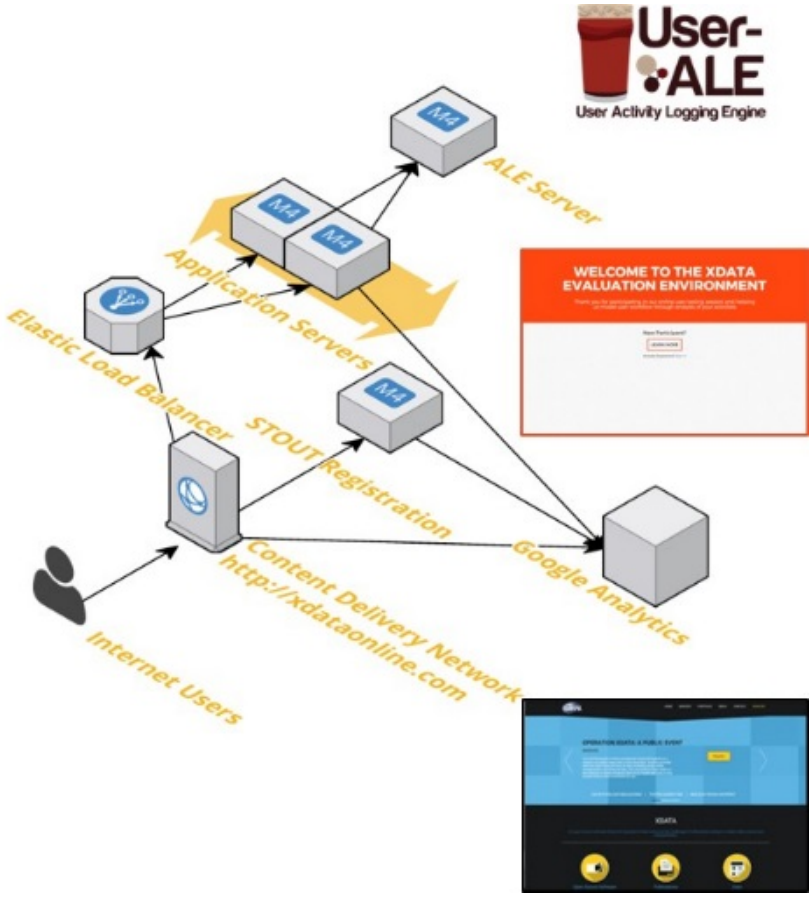


Figure 3. XDATA MOT Infrastructure.

Data was collected through Amazon's Mechanical Turk (MTURK) panel service. Roughly 1,180 MTURK users participated in the XDATA MOT. Of these participants, roughly 800 were suitable for analysis due to non-completion of measures, drop-out or other reasons.

3.2.3. User Activity Logging Modeling and Metrics Validation. In selecting and developing modeling approaches to understand human use of analytic software applications, Draper reviewed the extant research literature (and existing commercial offerings), identifying key gaps in the insights gathered from similar approaches. As a result, the following assumptions guided our efforts in developing modeling and descriptive metrics for evaluation purposes:

- Models for human usage would need to incorporate both how users allocate effort across the features of applications (e.g., application space), and the temporal patterns with which users work with applications (e.g., application time). Only by combining elements of "space" and "time", would we be able to sufficiently model workflow in a disaggregated way.
- Models for human usage would need to incorporate methods that allowed us to model canonical tool use, not just single use sessions, or particular users.
- Models would need to produce output that would be easy to incorporate into visual analytics or visualizations.
- Metrics would need to describe model characteristics that are intrinsic features of all models of different applications, so as to make one model comparable to others across the same metrics, with a similar interpretation.
- Metrics would need to be intuitive in their meaning, such that they directly indicate some key pattern of life or some strategy of use. Thus, validation of such metrics would provide tests of key behavioral patterns rather than a strict "data mining" activity for numbers that correlate with validation criterion.
- Metrics would need to be validated against traditional or "state-of-practice" measures, so as to not only verify that they contain the same (or more) information as traditional metrics, but also validate their utility and sensitivity against "'state-of-practice" measures as benchmarks.

3.2.4. XDATA Application Evaluation. Part of Draper's role on XDATA was to use both "state-of-practice" methods and novel methods and apparatus developed in course of the XDATA program to evaluate other performers' products. In serving in an evaluation role, we made the following assumptions:

- Comparisons between applications would utilize metrics and methods designed to ensure that applications were compared at a level of abstraction such that meaningful differences between applications could be ascertained; applications would be compared against metrics that are equally informative for all applications (e.g., "apples-to-apples", not "apples-to-oranges" comparisons.
- Evaluation activities would coincide with metric validation activities for novel models and metrics, so as to provide meaningful, scientifically rigorous context to performers and programs personnel so that novel metrics are interpretable in relation to "state-of-practice" methods and metrics.
- Steps would be taken to invite fair comparisons of applications wherever rankings were produced. This includes normalization or weighting techniques to ensure that quantitative differences reflecting the sensitivity of measures is taken into account prior to making judgements about the ranks of applications.

3.3. Publicity and Community Development

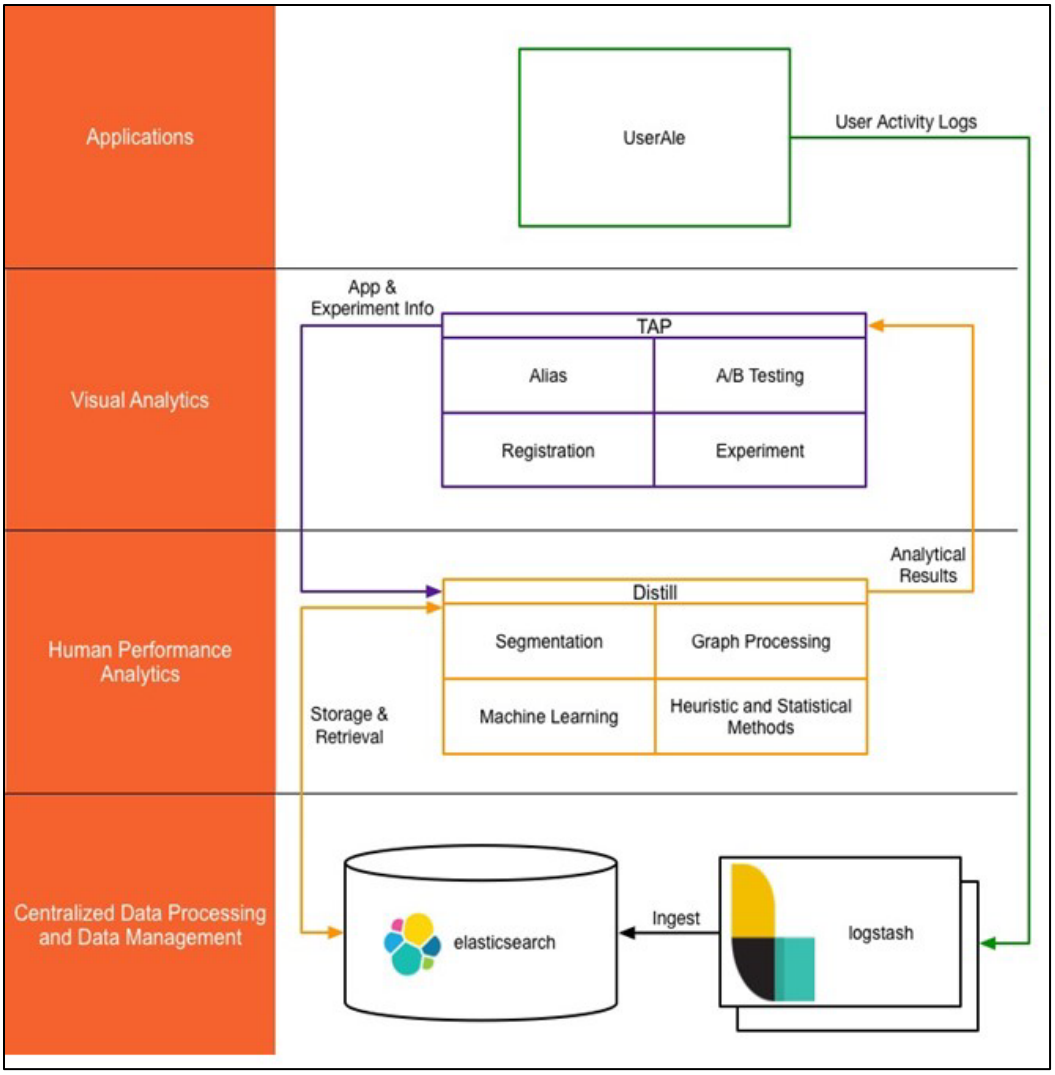
In the fourth year of the XDATA program, Draper worked to foster a community of interest around our open-source software as a transition activity. This community was to include both end-users of the technology, as well as prospective contributors—people that would want to help grow our technology. In order to accomplish this, Draper engaged in a number of outreach, marketing and community activities, operating under the following assumptions:

- Draper's Software as a Sensor™ open-source offerings would be more successful as community tools if they were included within a larger interconnected community that had an existing brand and public following (e.g., Apache Software Foundation).
- Draper would need to engage the public directly in marketing Software as a Sensor™ open-source offerings through public showings, exhibitions, and lectures. These activities would need to broadly canvas prospective users, including commercial, government and academic, and would need to coincide with industry meetings, conferences, tradeshows, etc.
- The funding provided for "productizing" Draper's Software as a Sensor™ open-source offerings in the fourth year would likely be sufficient to produce "minimally viable products"; for widespread adoption, enterprise level maturity would be necessary in order to accelerate and maintain a community of interest. As such working with US government end-users in "transition" activities would provide a means to identify and develop road-maps for gaps in Software as a Sensor™ open-source that would need to be addressed to make them viable to large enterprises.

4. RESULTS AND DISCUSSION

4.1. Software Development

Across 4 years as a performer on the XDATA program, Draper has imagined, innovated and engineered a viable open-source system for collecting user activity data, analyzing it for insights related to how usable and adoptable productivity (and/or analytic) software is. In year 4, Draper completed developments of an integrated Software as a Sensor™ system (Figure 4), and transitioned the open-source project to the ASF and DoD transition partners at NGA. Developments on the components of the system are described in subsequent sections, the system itself is deployable as an integrated whole through a pre-configured Docker container. Additional information about the project as well as all deployment and build documentation can be found at http://SensSoft.incubator.apache.org



- User ALE allows for collection of user activity logs through software tools.
- TAP allows users to register tools, control access, and supports visual analytics.
- Distill is an analytics stack that produces statistics, modeling output, and metrics
- System relies on searchable database for fast indexing and reliable data retrieval.

Figure 4. The Apache Software as a Sensor™ System.

4.1.1. The Apache User Analytic Logging Engine (ALE) product. In May 2015, we deployed UserALE v3 and began assisting developer in implementing it for User Testing and Online Testing. When users interact with specific elements (e.g., button, map, table) of an application, UserALE code injected into the source code related to those elements package and send a light-weight Java Script Object Notation (JSON) message detailing which class of event (e.g., Click, Hover, etc.) activated the User Interface (UI) element. UserALEv3 was a dramatic improvement over UserALEv2, and UserALE, which were simple prototypes (Year 1 and 2 outputs). Much of these improvements were related to making it easier for developers to apply labels to the API. Rather than mapping UI elements to specific kinds of analytic workflow components (e.g., Explore Data, Create View, etc.). These workflow components were derived from year 1 qualitative studies with analysts. However, in implementation developers had difficulty fitting these labels in mutually exclusive ways to their UI components (see Figure 5).

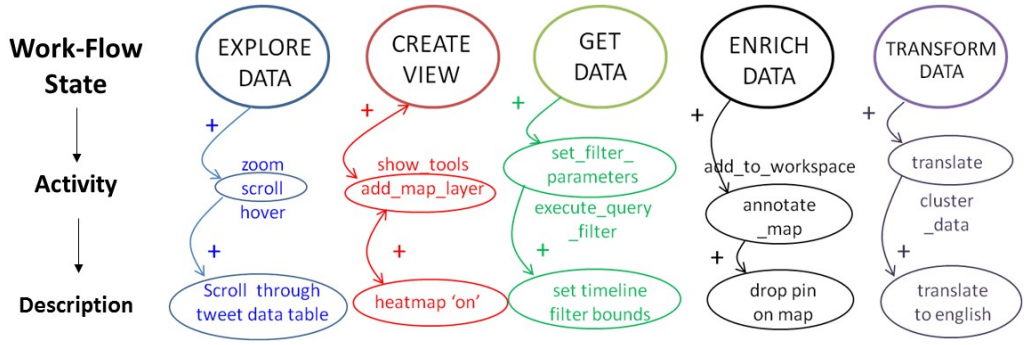


Figure 5. UserALEv2 Logging Schema.

UserALEv3, in contrast, adopted a ground-up model, letting developers label the elements and element groups of their application as they felt most appropriate. UserALEv3 information was very close in kind to Google Analytics, Adobe Ominture, Piwick and similar (Figure 6).

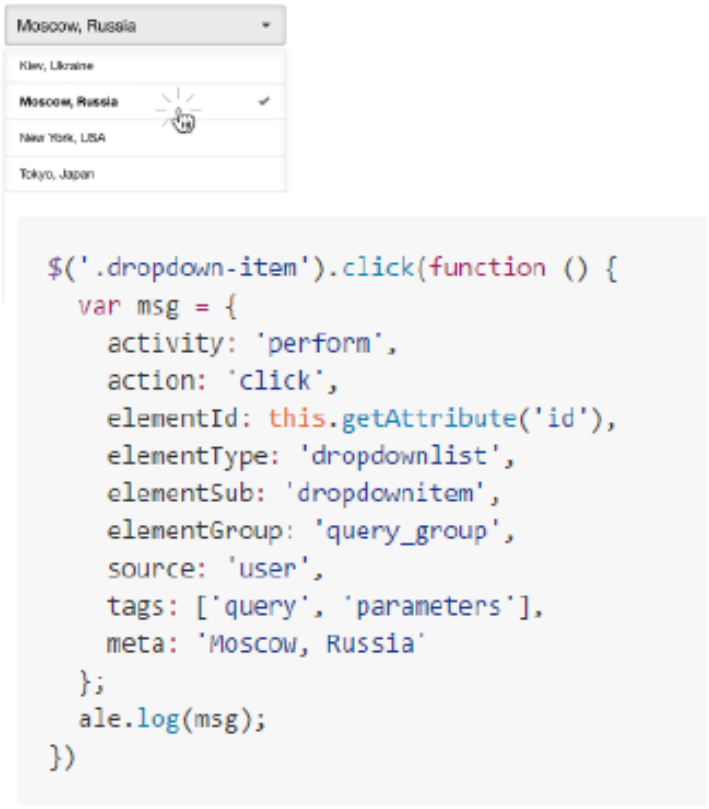


Figure 6. UserALEv3 Logging Schema as Depicted in JSON format.

Another major upgrade was the integration of ELK Stack (ElasticSearch, LogStash, Kibana), which improves our capabilities for rapidly collecting, archiving and accessing activity log data. Integration with Kibana provides an agile, configurable set of dashboards that developers (and programmatic personnel) may configure however they like to examine net usage of applications as well as what features of the applications they are using most. Kibana is also immensely useful for assessing how effectively UserALE was implemented, both evaluation and developer roles (Figure 7).

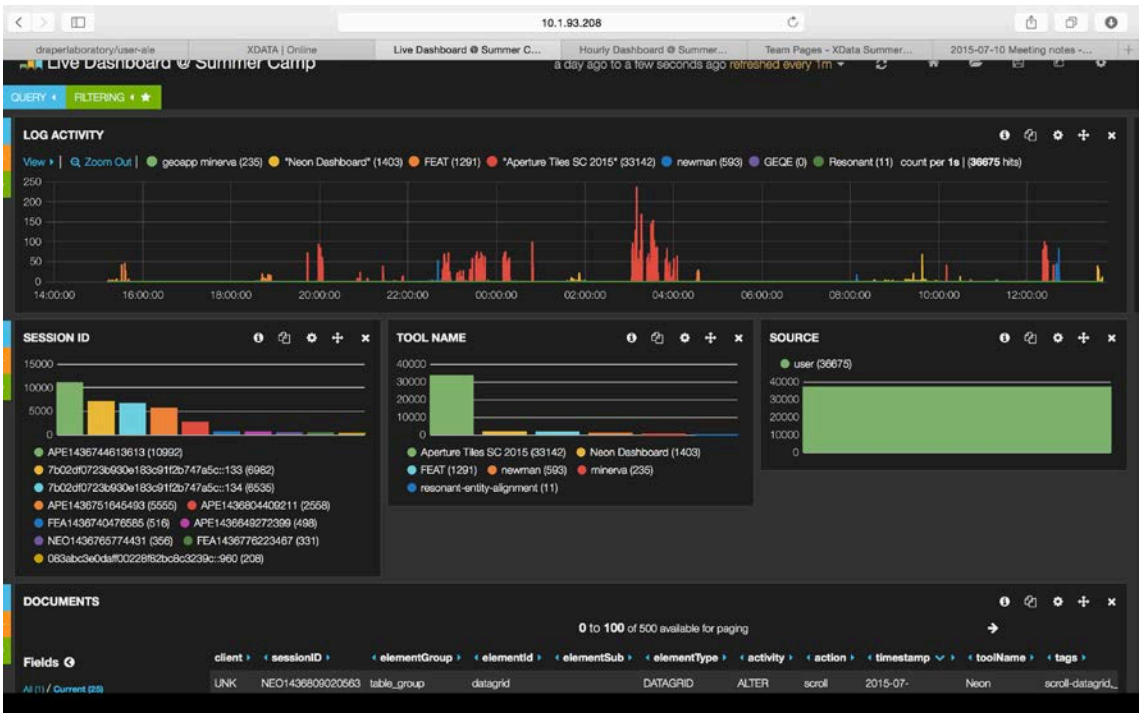


Figure 7. Kibana Dashboard for Visualizing Application Use.

In May 2016, Draper released UserALE.js, culminating in a viable productized version of our logging apparatus. UserALE.js has a number of improvements upon UserALEv3, including a dramatic reduction in level of effort (LOE) for deploying UserALE.js. UserALEv3 required developers to manually apply “hooks” for the UserALE service API throughout their source code. UserALE.js removed this dependency, reducing instrumentation to the application of a single line of code, injected into the top of the source code. This code is a “script tag”, which calls UserALE.js that exist outside the application. When called, UserALE.js initiates messaging services, recording the event received by event handler, the target element that received the event, the nesting of that target element within the applications’ Document Object Model (DOM) branching structure, cursor screen coordinates (x,y), and other configurable parameters (see Figure 8). In this way, UserALE.js is directly competitive with modern commercial services in terms of implementation, and is superior with respect the granularity of data contained in each log. This provides sufficient granularity for signal processing in a way that is reduced-to-practice and enables detailed workflow modeling as described in later sections. A persistent, live demonstration of UserALE.js is available at http://senssoft.incubator.apache.org. Not only does UserALE.js dramatically reduce the LOE for deployment, but enhances the richness in the data that the UserALE service provides for user activity gathering in web applications.

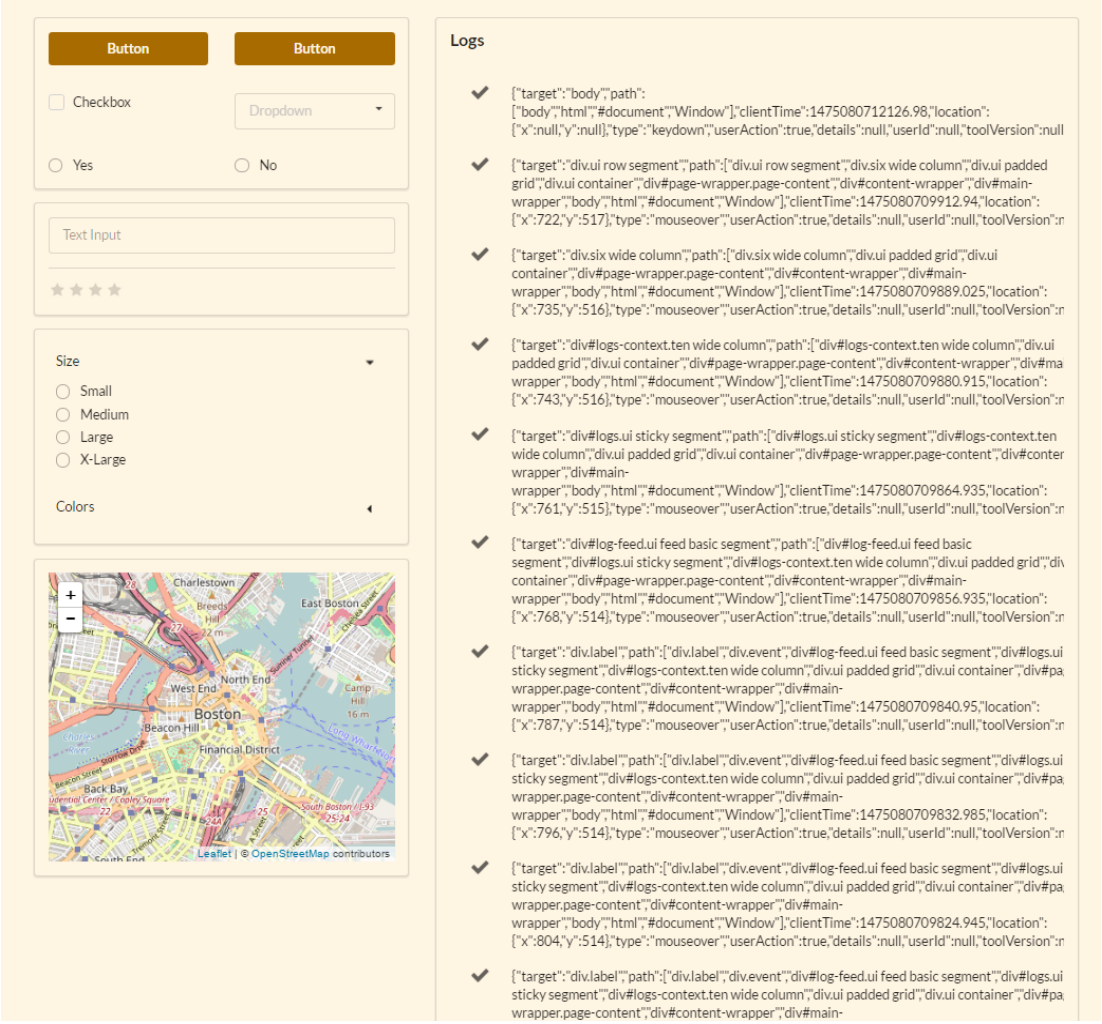


Figure 8. UserALE.js Demonstration and Log Structure.

4.1.2. The Apache Distill Product. In year 3, Draper realized that one of the most important user sets of usability findings are software developers themselves. Draper therefore began developing concepts and software for developers to interface directly with UserALE data, analyze it themselves and visualize it in distinct ways suited for developers. Our concept for Distill changed dramatically from year 3 and year 4, but the underlying theme behind Distill is to provide a direct interface between developers and UserALE data.

In year 3, we envisioned Distill as a simple web application to clearly and concisely deliver usage analytics. It visualized results such as sub-sequence metrics and demographic data in a way that enables developers to explore and understand how people are using their applications. This would allow developers to build a better understanding of their users, identify actionable insights, and ultimately iterate and improve their applications. In year 3, we thought to provide developers direct access to their users' sequential behavior, visualized as clusters of sequential behavior, with weights applied to how frequently users were observed within those workflows. These insights were observable through a

“sunburst” plot. This version of Distill thus attempted to provide developers a clear way to understand how their users integrated features in time (sequentially) to perform tasks. Developers could then subset user sets to explore how different classes of users made use of the application, for added insight (Figure 9).

Figure 9. Early Distill Concept Interface (Year 3).

In year 4 of the XDATA program, we completely re-engineered Distill in the service of productization. Rather than a user interface, we re-imagined Distill as a data interface and analytics stack, or framework. As the analytic framework of the Software as a Sensor™ Project, it provides segmentation, statistical packages and graph analysis for describing users’ interactions with the application to adopters. Distill is written in Python and utilizes packages like Continuum Analytics’ sci.py and num.py packages for statistical processing, and open-source packages like NetworkX for producing graph models. Distill is engineered so that certain aspects of processing can be off-loaded outside of Distill’s own Python environment, so that developers and data scientists can work with their data using their own analytics environment (e.g., Anaconda), but retain the ability to connect to UserALE data and Distill’s segmentation functions. Distill provides an interface directly into UserALE log databases, providing representational state transfer (REST) interfaces and structured query services for calling data, and applying a variety of models to that data. The segmentation feature allows developers or data scientists to focus their analyses of user activity data based

on desired data attributes (e.g., certain interactions, elements, etc.), as well as attributes describing the software tool users, if that data was also collected. Distill's usage and usability metrics are derived from a representation of users' sequential interactions with the application as a directed graph. This provides an extensible framework for providing insight as to how users integrate the functional components of the application to accomplish tasks. Figure 4 provides a schematic for how Distill connects and serves other Software as a Sensor™ services.

4.1.3. The Apache Test Application Portal (TAP). After the reimagining of Distill in Year 4, Draper developed a service for visualizing data processed through Distill, and a user interface (UI) for making queries against Distill for data and post-processing of that data for visualization. TAP was developed to fulfill this need, with additional features to provide a variety of functional front-end capabilities to developer end-users.

TAP is designed to be a scalable visualization platform to support the open-source Apache SensSoft project community. In this respect, adopters of TAP will be able to configure how various filter settings connect with their unique data structure, and add visualization assets for viewing their data as they see fit. In this way, development efforts emphasized integration of TAP with the rest of the Apache SensSoft project components via RESTful interfaces (see Figure 4). In order to serve visualization and integration needs, we developed TAP using Python and Django. This provides some useful features for user management, as well as a customizable environment that developers can use to embed visualization assets from open-sources like Data Driven Documents (D3) and REACT.

Draper focused its development efforts for TAP visualizations on unique features that were not otherwise included in other packages. For example, for users that are simply interested in seeing count or frequentist data on how many UI elements were interacted with or number of users, they may simply benefit from using Elastic's Kibana dashboard. Moreover, features like these are simple to "drag and drop" and configure in TAP using open-source assets like D3 (e.g., histograms). As such, Draper developed an interface that highlight the discriminating capabilities of Apache SensSoft, including graph analytics (via NetworkX libraries) for workflow models and version-to-version testing, i.e., A/B testing comparisons of activities from one version of an application and another (Figure 10) (via Continuum statistics packages like SciPy, and NumPy). Among our key innovations in visualizations is a customized chord chart with integrated, interactive visualizations for graph metrics (the "Bowie" plot) (Figure 11).

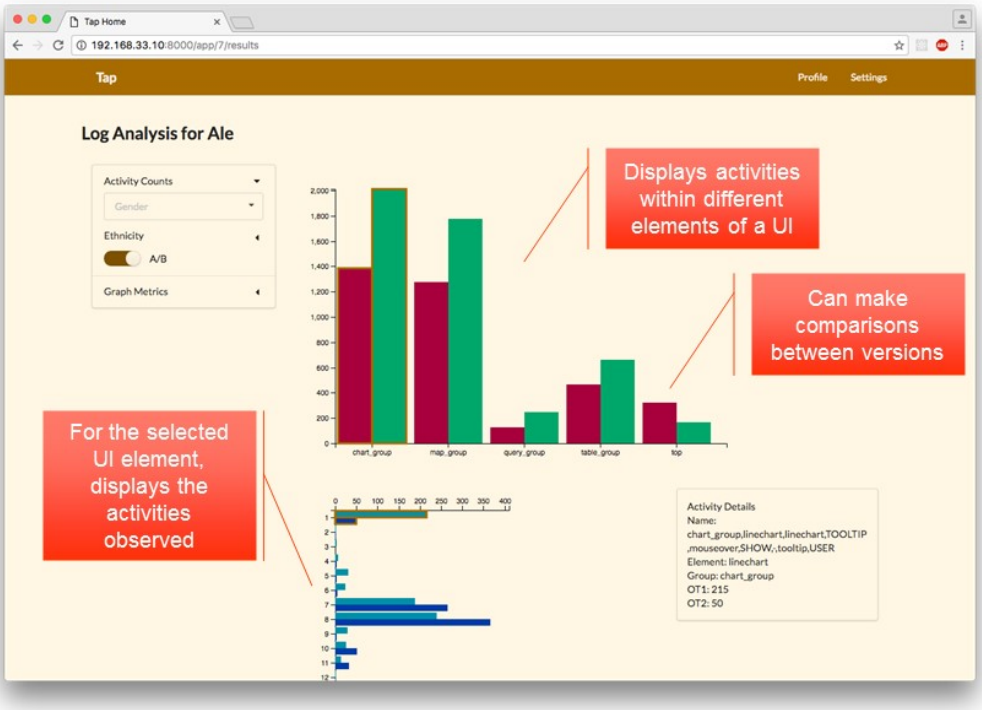


Figure 10. Apache TAP Dashboard for User Activity Analysis.

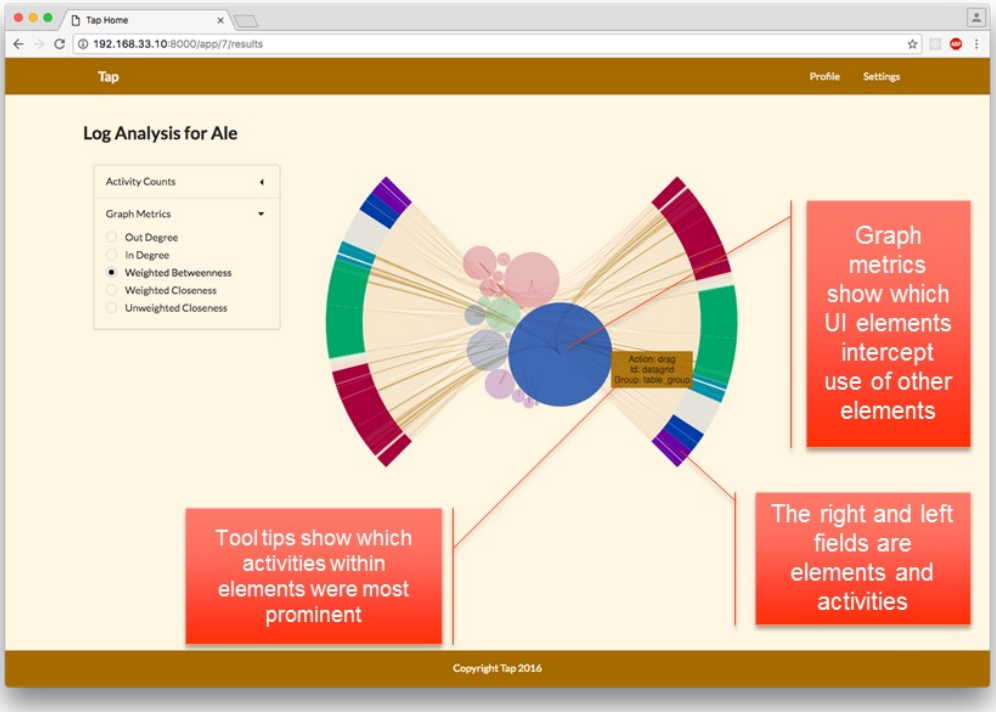


Figure 11. Apache TAP Dashboard for Visualizing User Workflows.

TAP's features for supporting the procedural aspects of user management are built on top of Django. Django is a Python development framework that makes it easier for others to make and add functional modifications. One of the key features leveraged for TAP, in this respect, is Django's ability to negotiate individual user permissions (e.g., registration, access controls, and persona management). This allows each adopter of TAP to register their own account within an instance of TAP, customize their views and utilities, independent of how other adopters might use it (e.g., persona management). However, TAP supports an information model that allows user to register as children of a parent organization, providing a means for sharing access to data stored in Elastic databases at an institutional level, and modifying these views at the individual user level.

4.1.4. The Apache Subject Tracking and Online User Testing (STOUT) product. The Subject Tracking for Online User Testing (STOUT) application was a prototype in year 2 and was developed into a functional suite of software in year 3 to assist with various human subjects data collection activities. STOUT's base functionality enables managing user pathing through experimental tasks (Figure *12*), and providing an instrumented interface for users to start and move through tasks presented through web forms (e.g., SurveyMonkey) in a way that is non-invasive with respect to access to a test application and the performance of that application (Figure *13*). Notable improvements include an improved user authentication system, the addition of experiment and achievements models to improve user tracking and engagement, a simplified process for registering new tools, and a UI refresh. In addition to design and implementation improvements on the STOUT system, we developed a larger ecosystem for integrating analytics.

The evaluation ecosystem can be seen in Figure *14* below and shows how STOUT integrates with the analytical tools SurveyMongo, Scale Computation and Codebook Handling (SCO+CH), and Distill. STOUT sits in the center and functionally coordinates the entire data collection and analysis pipeline through a RESTful API. STOUT, SurveyMongo, SCO+CH (see Figure *14*), and Distill are all products developed during year 3.

STOUT collects, organizes and stores metadata about participants along with task assignments and progress metrics. It is a content management system for formal human performance experimentation. STOUT presents tools and operational tasks to each participant. It tracks progress on an intake questionnaire to assess the user's background and experience as an analyst, and operational task performance. The STOUT system provides human-subjects testing experiment administrators with a flexible tool for managing and tracking user progress through a series of online tasks. The STOUT system was designed to help collect information about the utility of online applications.

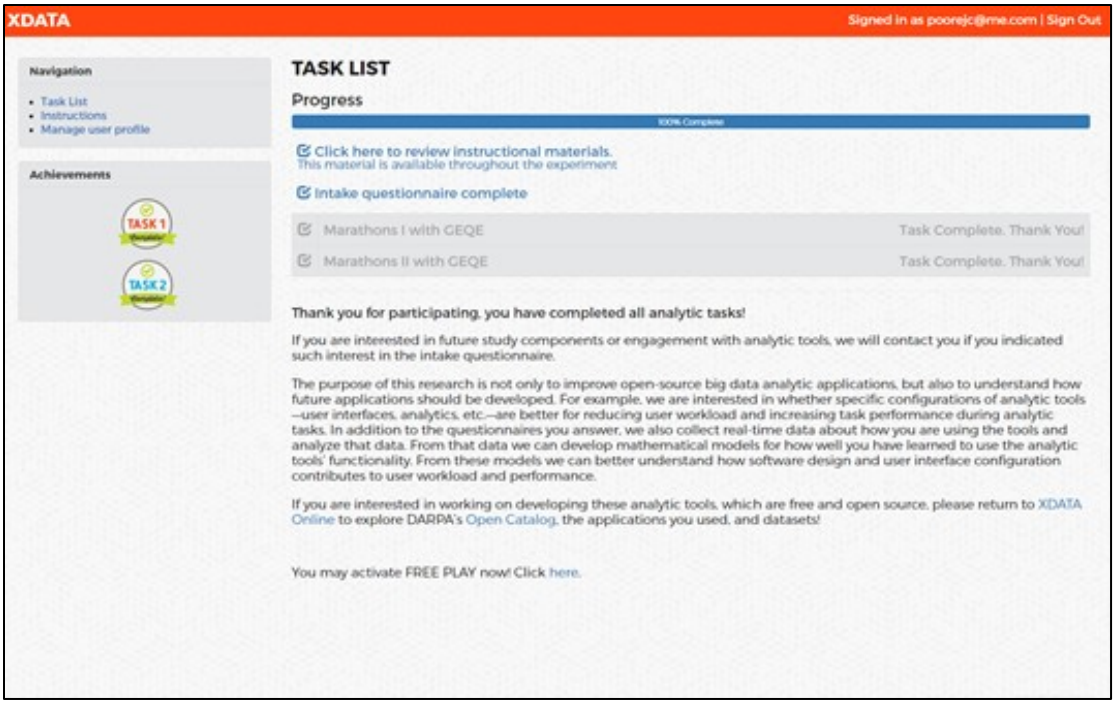


Figure 12. STOUT Participant Task Pathing Display, with Achievements.

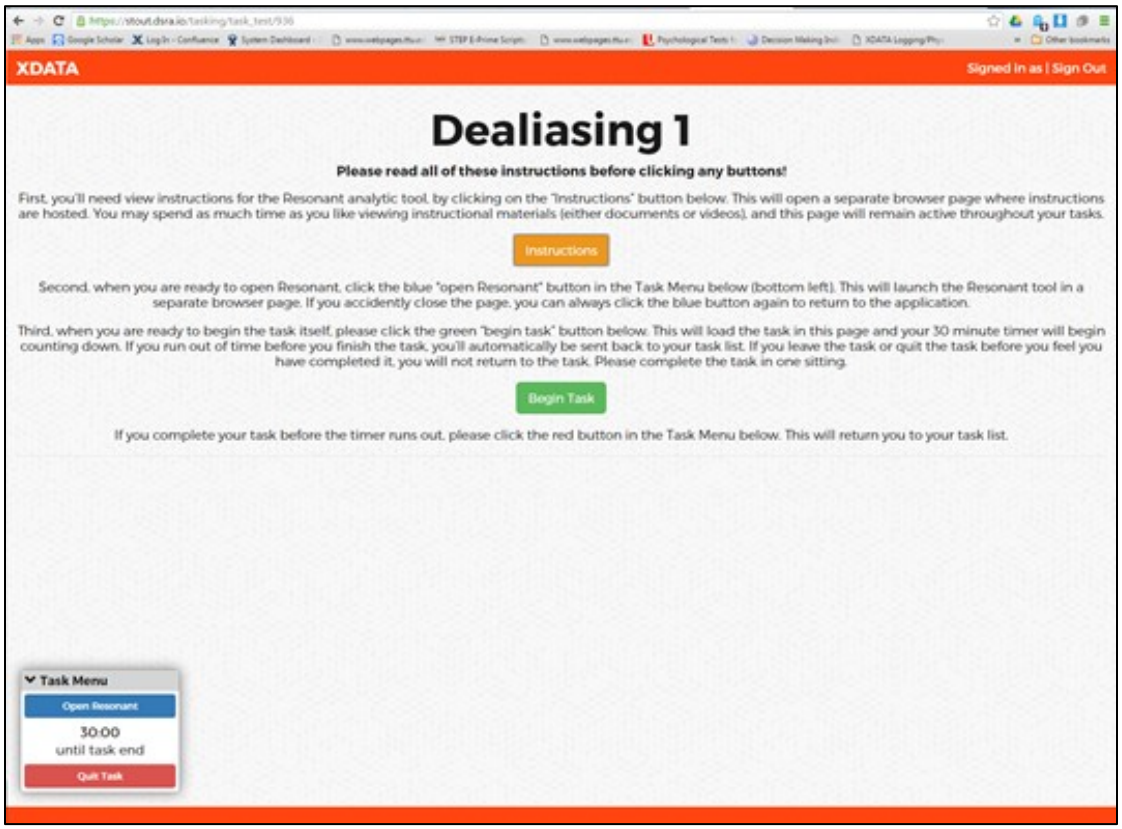


Figure 13. STOUT Task Portal.

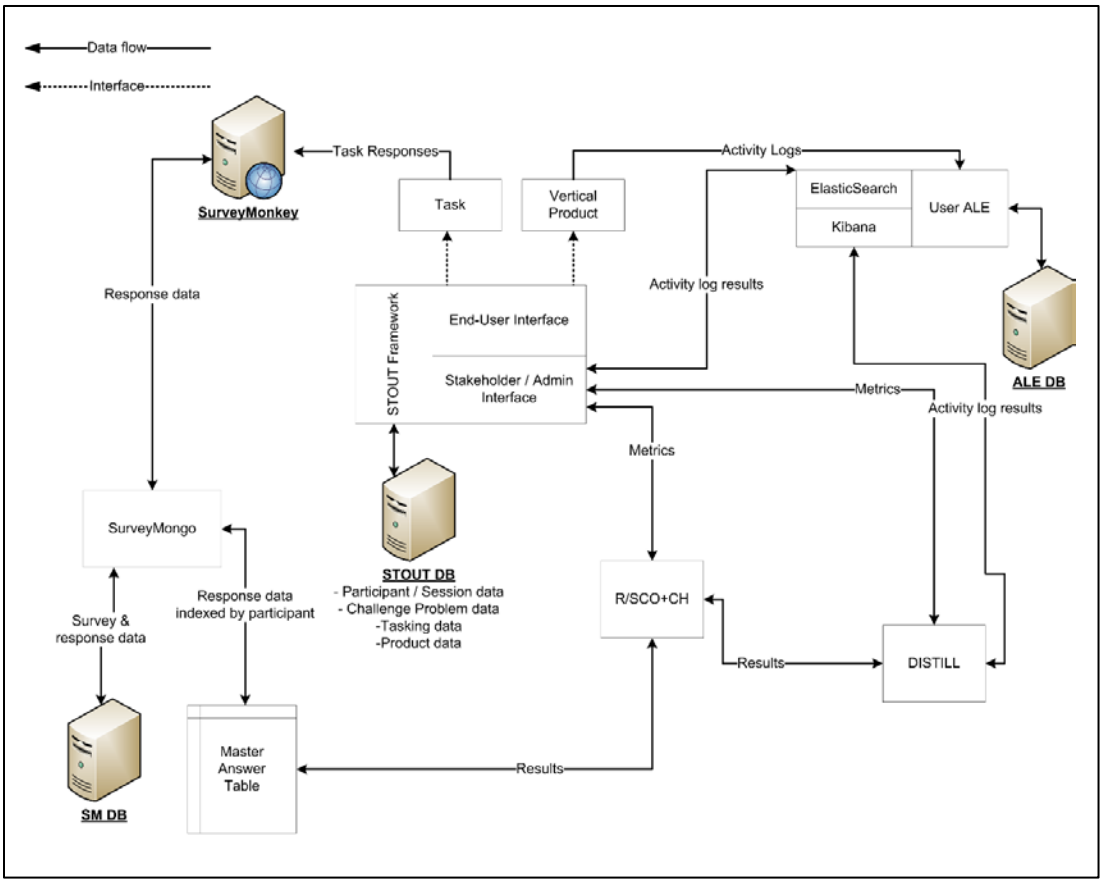


Figure 14. Draper's Evaluation Ecosystem.

STOUT is a web based database application built using the Django framework. The data models can be seen in Figure *15*. STOUT assigns registered users a user profile that is associated with their account and used to track their progress through the experiment. For year 3, the user authentication system was upgraded to include Django's built-in authentication authorization system which provides individual and group access controls to restrict navigation. New administrative features for managing content were implemented to provide additional enforcement of authorization protocols.

The remaining STOUT data models in Figure *15* provide content tracking and management throughout the experiment. A registered user is assigned an Experiment object that dictates the number of tasks to be completed, the time allowed for each task, whether or not these tasks should be done sequentially or not, and if the particular experiment requires an intake and post task questionnaire. Individual tasks are managed with the Task List Item object that provides an association between a User, a Task and a Product (tool). Tasks and

Products are associated with one another through a common Dataset (challenge problem). A final data model was added for Achievements to further incentivize participation by rewarding top performers or active participation. Achievements can be earned for accuracy, completion time, and peer referrals.

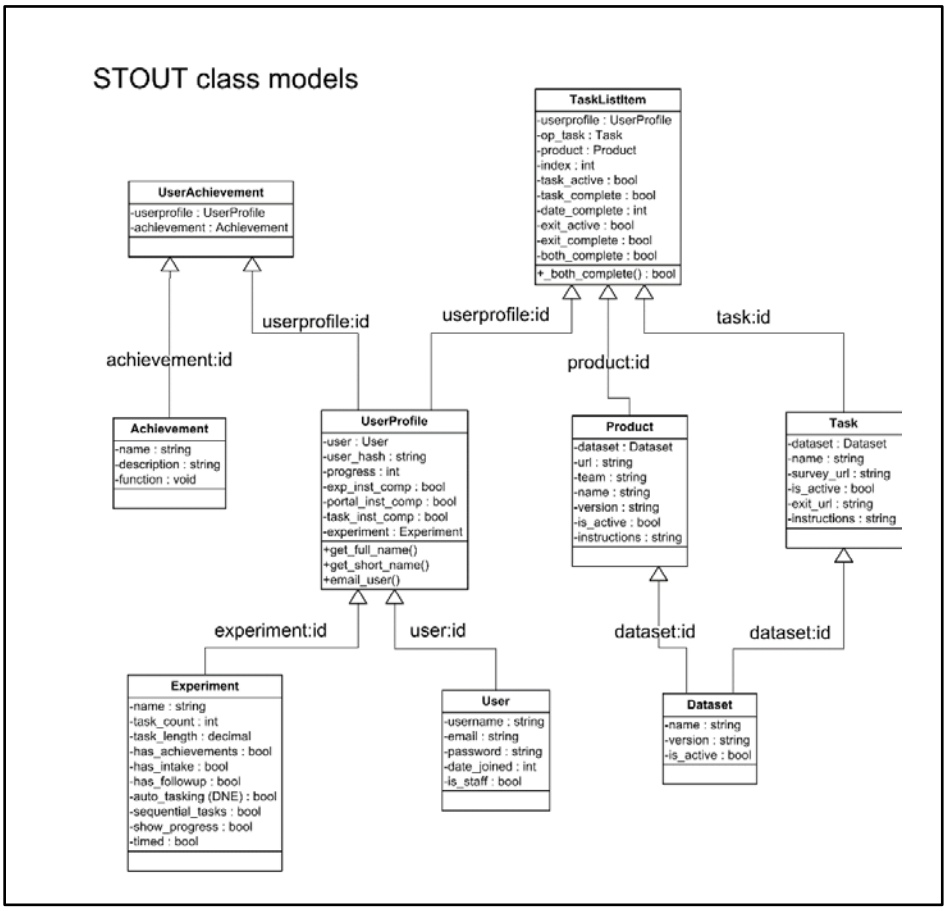


Figure 15. STOUT data models and their organization.

A group of administrative controls were added to STOUT for year 3. These controls simplify the addition and management of all the data models within STOUT. Now, any user with the proper administrative authorization can add or manage products (tools) and their associated tasks. New users can be registered by administrators prior to their participation to facilitate the experimental process. Task and product (tool) assignment can also be managed directly from the experiment administrator portal (Figure 16). Additional features for the experiment administrator through the new portal include viewing results generated from apps connected on the analysis pipeline (see Figure 17).

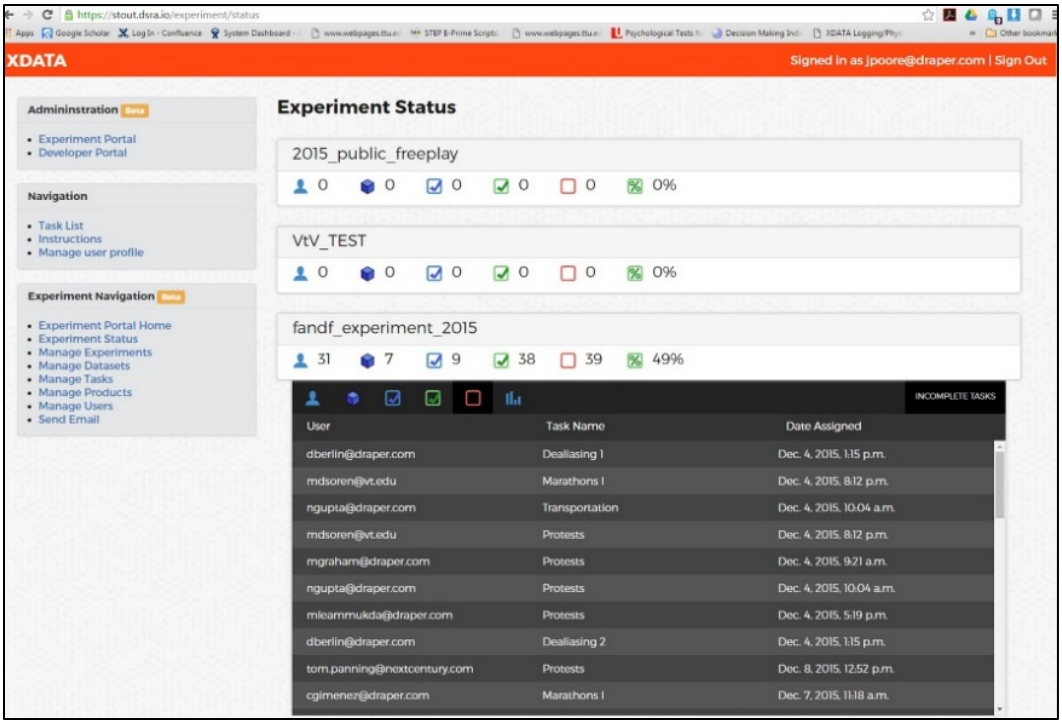


Figure 16. STOUT Participant Administration/Management Features.

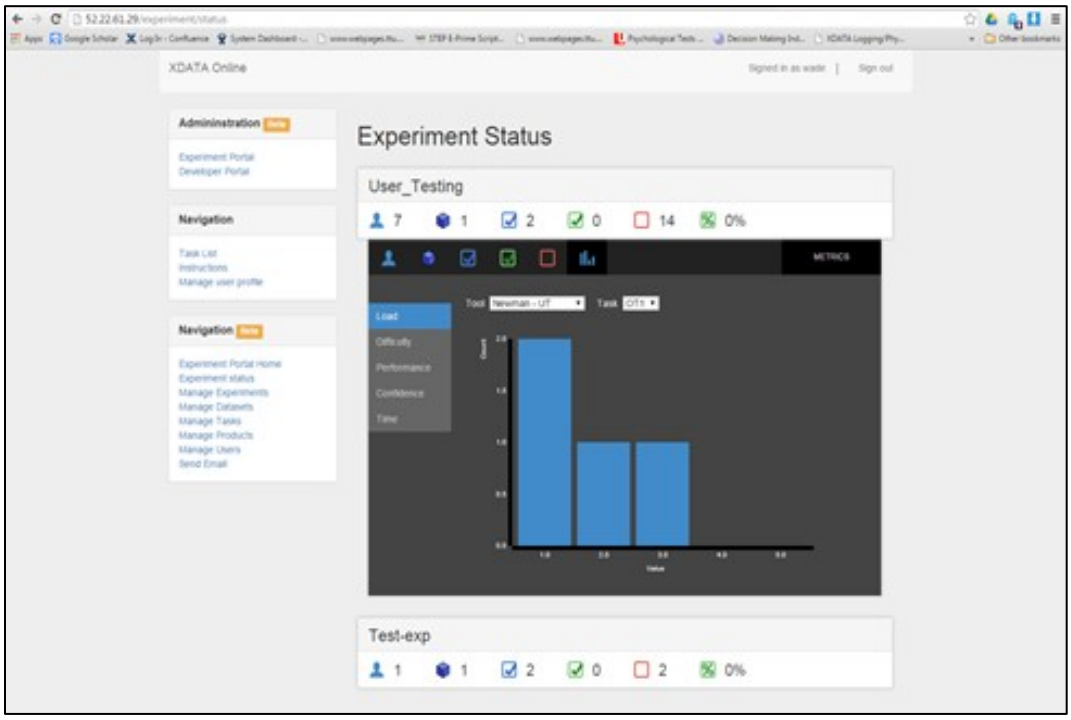


Figure 17. STOUT's D3 Visualization of Processed Data.

In preparation for the massive online experiment hosted at the end of year 3 and during year 4, a number of security features were added to STOUT, and a

number of vulnerabilities were tested. Figure 18 summarizes a comprehensive list of these security tests performed to maintain data and application security during online testing.

XDATA Security Topics			
General Security			
Topic	**Vulnerability / Risk**	**Complete**	**Dependency**
Limit admins	More accounts for potential unauthorized access	Yes	Internal
Limit real-time updates	Chance of inadvertently taking out existing functionality	Yes	Internal
Use current versions	Prevent usage of modules with known security risks.	Yes	Internal
Secure code storage	Unintended visibility or execution of Python code	Yes	Internal
NGINX configuration	Set up general security settings on NGINX installation	Yes	Internal
Database			
Topic	**Vulnerability / Risk**		
Secure user input	Prevent SQL injection attacks	Yes	Internal
Backup database	Loss of data if corrupted or accidentally deleted.	Yes	Internal/L-3
Secure Postgres	Unauthorized users gaining access to DB contents.	Yes	Internal
Encrypt database	Unauthorized users gaining access to DB contents, even if encrypted.	No	Internal
Password complexity	Prevent easy password cracking	Yes	Internal
Data exfiltration prevention	Data leaving database unnoticed	No	Internal
Protect PII	Loss of privacy for user group	Yes	Internal
Secure user credentials	Loss of privacy of user credentials	Yes	Internal
Separation of resources	Accessing DB through web app	Yes	Internal
Web Application			
Topic	**Vulnerability / Risk**		
Web Application Firewall (WAF)	SQL injection, XSS, CSRF, etc.	Yes	Internal
HTTPS only	Passing unencrypted data such as logon credentials; MITM attacks	Yes	Internal/External
Secure session ID	XSS by guessing session IDs	Yes	Internal
Session Security	XSS by reusing session IDs	Yes	Internal
Limit User Sessions	Denial of Service attack through opening excessive sessions (Other DDoS risks, but this is a nice idea for preventing increased system load and one avenue to a problem.)	Yes	Internal
Auto-escape with templates	XSS in Django 1.7	Yes	Internal
CSRF protection	Prevent usage of one user's credentials by another user	Yes	Internal
Clickjacking prevention	Prevent malicious rerouting of users via content in hidden frames.	Yes	Internal
Older browser security	Browsers older than IE8, Firefox 3.6.9, Opera 10.5, Safari 4, and Chrome 4.1 won't prevent clickjacking with the above settings.	No	Internal
Authentication throttling	Avoid brute force attacks	Yes	Internal
Lock root account	Rogue user gets access to root account and has privileges to access entire server.	Yes	Internal
Prevent hanging sessions	User leaves browser open and unauthorized user uses account before the session expires.	Yes	Internal

Figure 18. STOUT security vulnerabilities tested and addressed in year 3.

SurveyMongo is a small scale application deployed with STOUT and enables STOUT to poll multiple data sources (e.g., questionnaire data, user registration data, etc.) (Figure *14*). It is a web based database application designed to automate and index task response data from participants and prepare it for analysis. The tasks that are presented through STOUT are created with surveys built at SurveyMonkey.com. These surveys are then presented to the user through an interface in STOUT. Every response that the user submits is appended with a unique hash created from the user's username to ensure anonymity but preserve traceability between users during analysis. SurveyMonkey.com provides an API to access responses and that API was used to build SurveyMongo.

The main requirement that drove the design and implementation of SurveyMongo was the need to organize data by participant. SurveyMonkey.com organizes response data by survey, but our analysis pipeline requires that data be organized by participant. SurveyMongo downloads response data for all tasks from SurveyMonkey.com and then re-indexes it by participant, combines it with logistical data from STOUT, and organizes it into a tabular format where each row is a unique user and the columns are all the features of interest. As seen in Figure *14* above, this table then used by SCO+CH to perform additional scale computations and append metrics to this table that can be used in analyses and viewable through STOUT.

SCO+CH is an R language script concept for automating form data (questionnaires, surveys, etc.) post-processing and managing codebooks for large research datasets. Particularly for datasets that need be shared across sites and those that might contribute to legacy datasets, it is imperative to carefully manage how raw data is combined or aggregated into composite variables that may then be used in statistical analyses and hypothesis testing. This is critical to ensuring that empirical findings can be reproduced and expanded upon and requires tracking how raw data is weighted, normalized, etc., before aggregation into composite variables. In reality, many researchers calculate and name composite variables for use in research using private schemes and code bases, in a variety of statistical (SAS, SPSS) and engineering tools (MatLab, Python). These formats and methods are not often interchangeable or interoperable, making standardization of methods and sharing research data difficult.

In preparation for the Massive Online Experiment, we began developing an R code base with functions designed to read and manage codebooks containing metadata about how raw data is aggregated into composites, scales, etc. SCO+CH ingests both raw data and a codebook, applying item reversals, weighting schemes, etc. SCO+CH capitalizes on a standardized variable naming convention to parse variable names of raw data variables and composites to be computed. Based on the structure of these variable names, SCO+CH will compute these composites using the correct raw data variables (Figure 19). SCO+CH keeps master data tables organized for shared use, and allows for scalable data collection efforts by preventing the need for “hard-coding” aggregation commands when new data is added to the raw dataset. So long as the raw data variable naming convention conforms to a certain scheme, new logic for computing new composites can be input using the variable of the desired composite.

SCO+CH is currently integrated into the Evaluation Ecosystem (Figure *14*) and computes metrics from questionnaire data and activity log metrics. While currently a series of scripts integrated into an application with a simple wrapper, we intend on developing it further and releasing it as an open source R library that anyone can benefit from. SCO+CH is integrated into the Online Testing Infrastructure as a script wrapped to function as a web app. In this way, other applications (e.g., STOUT) can initiate SCO+CH scripts, which then produces metrics and composite variables automatically (see Figure 19).

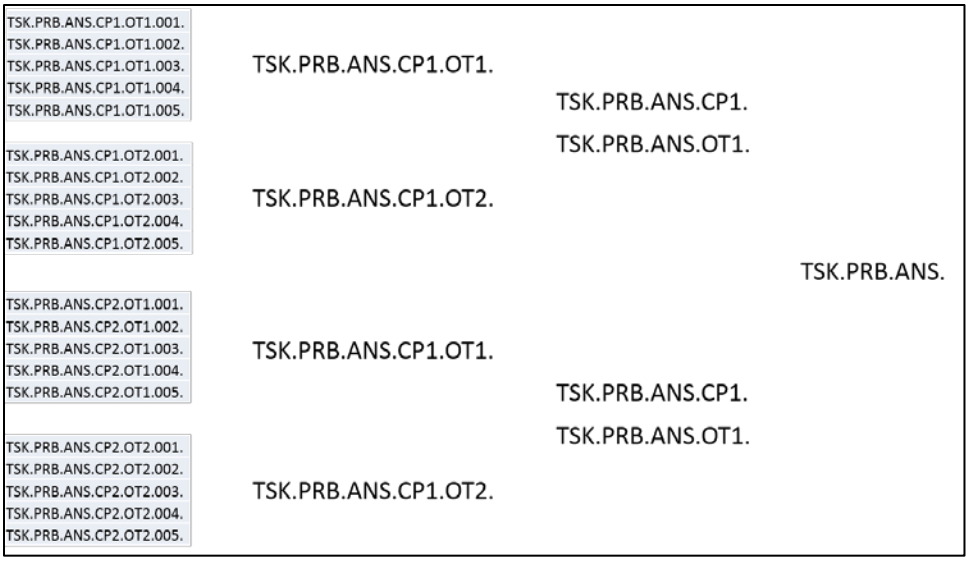


Figure 19. SCO+CH Aggregation Operations.

4.2. Evaluation and Human Subjects Research

An important aspect of Draper's role on the XDATA program was evaluation and human subjects research. These two activities worked hand in hand, as the challenge issued to Draper by DARPA was to innovate in methods for evaluating software. As such, as we collected data for evaluations, we used that data for developing new modeling approaches for user activity logs and validating quantitative metrics for use in evaluation.

4.2.1. Evaluation Events. Throughout the course of the XDATA program, Draper worked with DARPA to plan and oversaw 3 data collection events (years 2, 3, 4). Given the maturity of applications in year 2, rather than an evaluation, year 2 served as a means to demonstrate and validate initial capabilities in user activity logging and analyses of prototype applications. Evaluations were performed in both year 3 and 4, however, as DARPA issued guidance to performers to enhance prototype applications and productize them for public and USG consumption.

4.2.2. User Activity Logging Modeling and Metrics Validation. As part of our role on the XDATA program, we were challenged to create new, objective metrics regarding analytic application usability. In order to accomplish this we adapted modeling techniques developed on internal research and development (IR&D) funding, as well as innovated new approaches. These modeling techniques are designed to exploit the information contained in user activity logs (collected through UserALE products). Metrics that describe the properties of models developed were identified and then validated against current state of the art methods for ascertaining usability. Overall, validation research conducted in year 2 of XDATA, as well as research that coincided with application evaluations in

year 3 and 4 provides confidence that data we collected from applications (via UserALE products) contain objective usability information. This research also provides confidence that the modeling approaches we applied to that data also yield information related to usability. At each year we were able to replicate these findings, demonstrating a capability for providing objective, minimally invasive software usability analysis. Additionally, in each year, as we assessed more mature applications tied to more complex, real-world analytic concepts of operation (CONOPS), we showed generalizability evidence that our technology and modeling approaches are capable of adding value both in and outside of the laboratory—replication and generalization are the benchmarks of successful science and technology.

In 2013, Draper adapted Beta Process-Hidden Markov Modeling (BP-HMM) for use in modeling sequential behavior data from software activity logs [13]. A Hidden Markov Model (HMM) is a type of Markov model that represents the dynamics of a stochastic system as a set of states and state transition probabilities, where the states themselves are not observable. A data (observation) sequence is generated by an HMM according to the set of observation probability distributions associated with the hidden states, and the state transition probabilities of the model, both of which are learned from the data [14]. Hidden Markov models have been applied in many cases to understand sequential human behavior and workflows, such as how body movements are coordinated (from motion capture), goal-related behavior in robotics research, and even behavioral (gestural) software inputs [15-18].

A key shortcoming of traditional Hidden Markov models is that they can only represent the dynamics of a single underlying Markov process, and comparing between two different models is like comparing apples to oranges. If we were only interested in examining how one user performed tasks with a given software application, and the unique behaviors of that user, the standard HMM formulation would be sufficient. However, the goal of this analysis is to understand canonical behavior patterns in an entire *ensemble* of logs. To accomplish this, we implemented a version of the HMM designed for multi-sequence processing, the beta-process HMM (BP-HMM). The BP-HMM is a non-parametric, Bayesian implementation of Hidden-Markov Models (HMM) that has only recently been introduced as a means for understanding sequential human behavior (typically motion capture) [19]. We have expounded on the original BP-HMM formulation by improving aspects of its implementation, including development of an automated parameter selection process and heuristics for model selection that reduce subjectivity [13]. The result is a global library of software usage states—different ways that users combine software functionality to accomplish tasks—that represents all the behavior of all users observed with an application. Once generated, this global library of states provides a way to describe each user's session as a sequences of states and state transition probabilities, enabling identification of canonical software usage patterns for each software application.

Each state that emerges from activity logs through our BP-HMM implementation represents how different software activities were integrated in frequency and time in a way that is substantively different from other ways of integrating these activities (Figure *20*). BP-HMM is highly germane to the study of software usability, how users learn to use software, and whether software makes

it difficult for users to coordinate their efforts on tasks, beyond the difficulty of the tasks themselves (e.g., cognitive load). This is very appealing from a software evaluation perspective. However, while BP-HMM allows for generalized models across users of the same software, those models do not generalize across different software applications. In order to make comparisons of usability across different applications we developed metrics in 2014 for describing gross properties of these models to enable these comparisons.

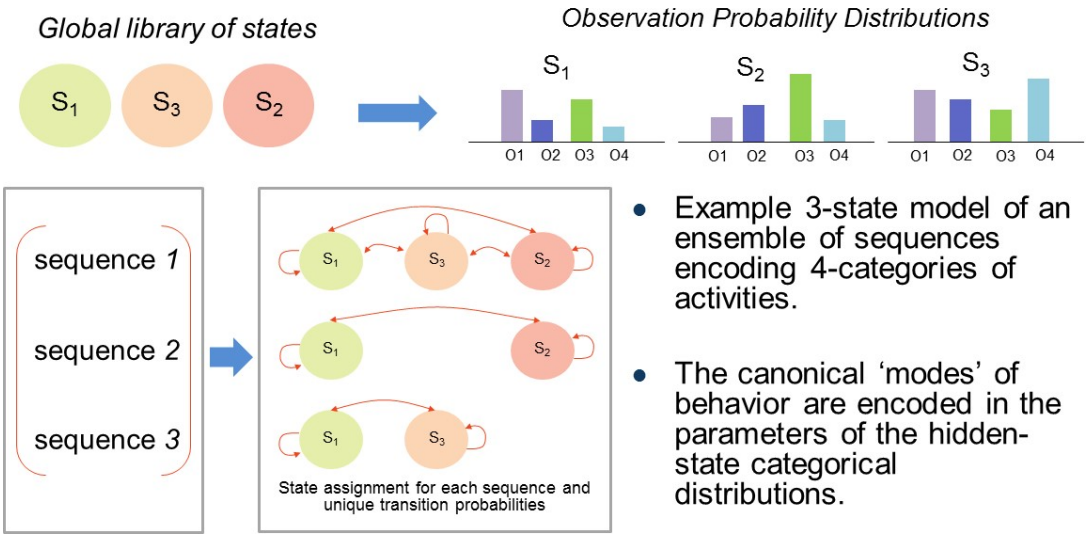


Figure 20. BP-HMM Approach to Modeling User Activity Logs.

Though BP-HMM states are different from one another, both within and across different software applications, they represent the same information—the likelihood with which software activities are observed together in time. Thus, while qualitatively different, the distributions of these probabilities can be similarly described quantitatively, agnostic to the meaning of those activities. For example, if users understand an application, we would expect them to understand how to use different UI elements and functions together, in an integrative way. In a BP-HMM model, this would be expressed in probabilistic distributions of activity that were more uniform or flat—diffused activity across the software's functionality. Alternatively, reliance on just a few functions might indicate a limited understanding of the software and would be expressed in more peaked distributions. These distributional shapes can be easily described with simple central tendency statistics (maximums, averages) and kurtosis (sample excess kurtosis) statistics.

Central tendency statistics (maximum averages of activity frequencies) can help classify different states as being uniform or peaked, given the unique state space of each application. Kurtosis or other measures of central mass in distributions allow for classifications based on objective comparisons against excess values with more rigid definitions. Metrics can then be calculated by establishing the proportion of time with the application that users spend in

uniform or peaked states [13]. These metrics, which describe integrated or dis-integrated use of applications, can be used to compare different software applications; our published work illustrates that the amount of time users spent in non-integrative states (e.g., “peaked states”) is positively correlated with measures of cognitive load [13].

Figure 21. BPHMM States Provide the Necessary Information for Deriving Integrated Use Metrics.

Across year 2, 3, and 4 of the XDATA program we have been able to replicate findings between our BP-HMM integration metric and subjective reports of cognitive load, task difficulty, and performance. The amount of time users spent in non-integrative states (e.g., “peaked states”) is positively correlated with reports of cognitive load and task difficulty, and inversely correlated with task performance. We also find (across the three studies) that metrics extracted from the BP-HMM approach outperform any that are calculated from raw software activity logs, such as activity rate/min, and are competitive with eye-tracking metrics.

Cognitive load and task difficulty were key metrics against which to validate BP-HMM metrics for integrated (or dis-integrated) use. Where task difficulty is the degree to which the task is intrinsically difficult, independent of the application (or tool) used to perform the task (Figure 22), cognitive load is the degree to which the application impedes task performance, independent of how intrinsically difficult the task is (Figure 23).

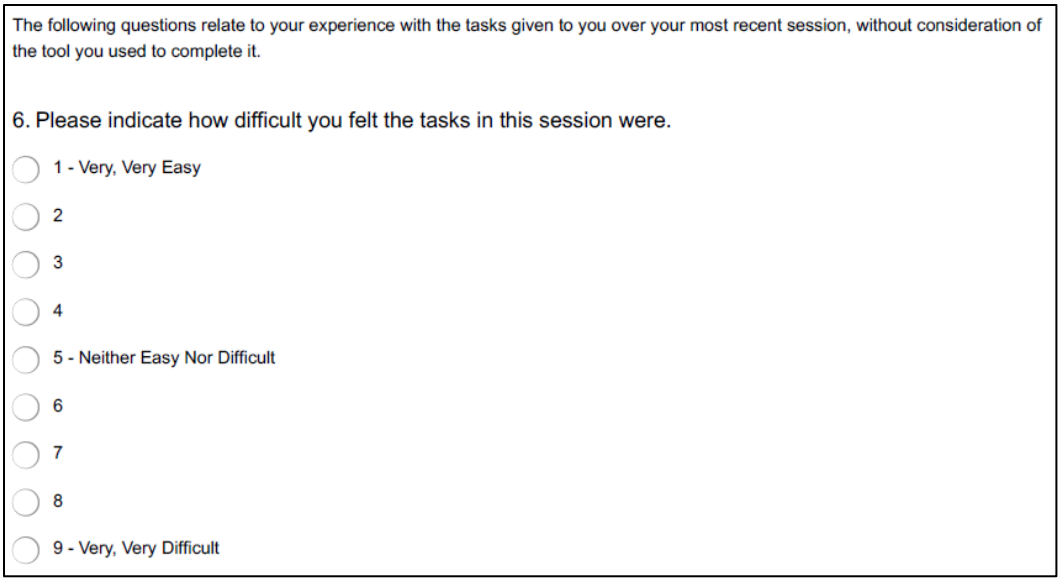


Figure 22. Task Difficulty Question from Post-Task Questionnaire.

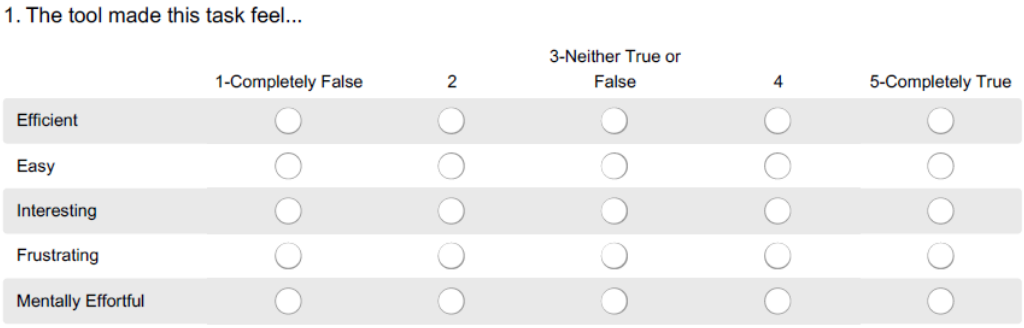


Figure 23. Cognitive Load Questions as Presented in Post-Task Questionnaires.

Across year 2, 3 and 4 of the XDATA program we were able to confirm that integrative use metrics owing to the BP-HMM modeling approach are correlated in meaningful ways with cognitive load and task difficulty metrics. In years 2 and 3, we were able to replicate findings from a pilot study completed under laboratory conditions using a simple, game-like interface (Figure *24*). In this study, we were able to show that the time users spend in "peaked", dis-integrated usage states is positively associated with reports of task difficulty ($r = .62$, $p < .05$; $R^2 = .38$). In year 2 of the XDATA program, we replicated this finding ($r = .58$, $p < .05$; $R^2 = .34$) using prototype applications developed by XDATA performers. In year 3, we not only replicated this finding, but generalized findings it to more mature, advanced analytic applications using large-scale datasets and operationally relevant tasks crafted in collaboration with former analysts ($r = .45$, $p < .01$; $R^2 = .21$). In year 4, we validated these effects at scale in a massive online testing events with MTURK users ($r = .42$, $p < .000$; $R^2 = .18$) (Figure 25). Across each of these years, we also found that these metrics were associated

with task performance (Figure 26). Additonally, in each year we found that BP-HMM metrics outperformed simple activity rate metrics that have become the industry standard for how to use user activity logs in workload analysis [20].

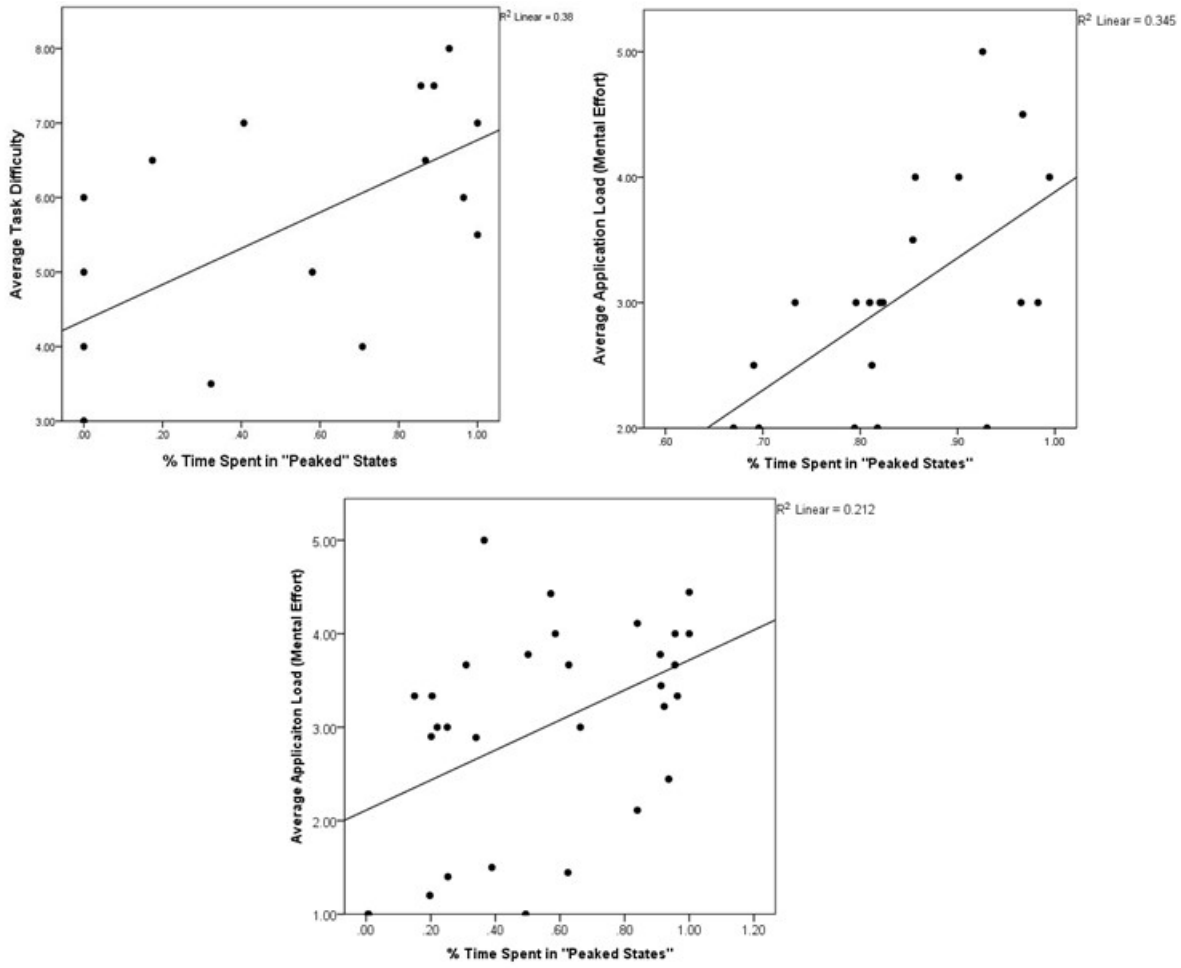


Figure 24. Scatterplots with Regression Lines Illustrating Association between Objective Integration Metrics and Subjective Cognitive Load Metrics.

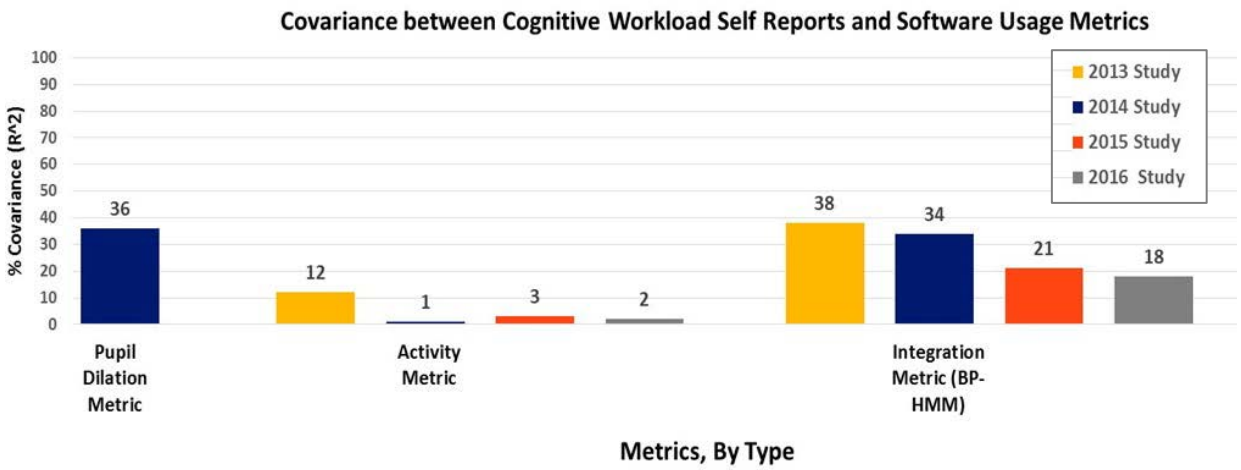


Figure 25. BP-HMM Integration Metric Effect Sizes in Predicting Cognitive Load, Compared to Other Metrics.

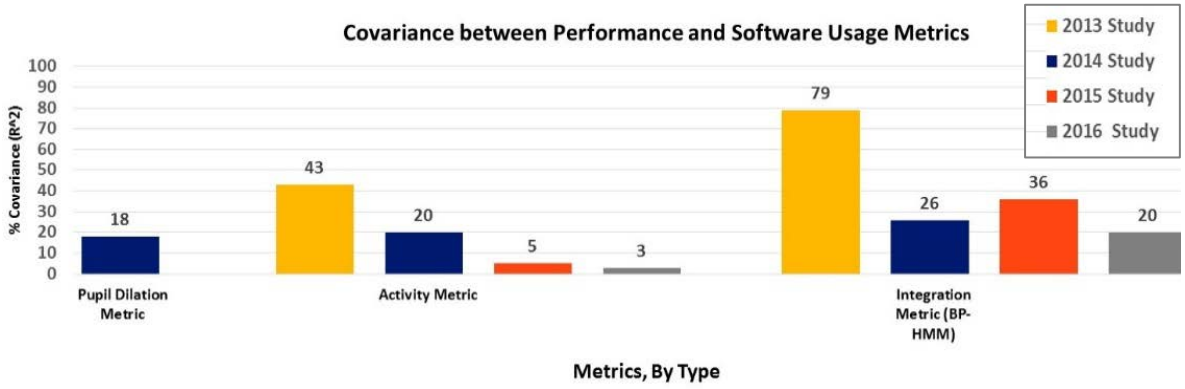


Figure 26. BP-HMM Integration Metric Effect Sizes in Predicting Task Performance, Compared to Other Metrics.

In year 2, we also assessed whether the BP-HMM modeling approach was sensitive to the same information as eye-tracking data, which is frequently used to ascertain application usability. In this analysis, we modeled a time-series of pupil dilation data around BP-HMM specific and non-specific features. We found that BP-HMM features (state-transitions) embedded in the pupil dilation time series accounted for more variation in pupil-dilation data than either non-specific features (activities) or "dead-space" (null-events) (Figure *27*). This suggests that BP-HMM metrics capture the same or similar information that is obtainable with state-of-the-art laboratory measures, at a fraction of the cost and with measures that are feasible in uncontrolled operational environments.

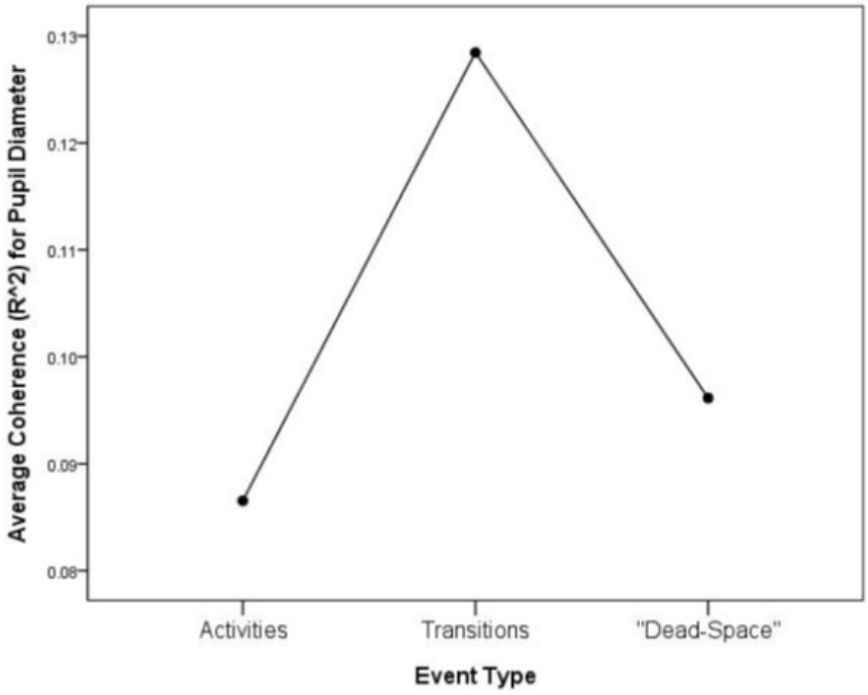


Figure 27. Relationship between BP-HMM Modeling Features and Eye-Tracking Features.

Our findings also suggest that it may matter less that applications have peaked usage states, or how many they have, but rather how much time users spend in them. Each application may have a strong random intercept with respect to the percentage of time that users spend in activities. This suggests that while each application may have different canonical usage patterns (more or less integrative or “kurtotic”), applications can be ranked based on how each application forces them into states that illustrate less integration across software functionality. We are currently examining these random effects with massive online testing data given the substantial statistical power we were able to accrue through sampling. Overall, our BP-HMM metrics have direct traceability to software functions that can be fixed, and our data is sampled from end-use; the metrics described above do not depend on any particular task structure, only that users have their own tasks. This is a major advancement in the fields of human computer interfacing and affective computing.

In years 2 and 3, we also experimented using sub-sequence modeling as a complimentary approach to BP-HMM to add granularity to our analyses. Sub-sequences are atomic-level workflows describing the patterns of activities that users integrate when performing tasks. We applied this approach to year 2 data and found that users who generated longer, repeated sub-sequences, performed better on tasks. We also realized that from a developer perspective, this highly granular perspective into usage is likely more useful than BP-HMM models for informing application improvement efforts. In year 3 we developed new methods

for extracting sub-sequences that are more germane to visualization and exploration. We also identified new quantitative ways of describing user's sub-sequences that while somewhat useful for evaluation, will prove to be more useful for future workflow modeling and visualization. Below we summarize our exploration of subsequences in Year 3 of the XDATA program.

For each application, the complete set of unique activity logs that could be generated through interaction with the interface was identified and used to create a master log code dictionary. Each log was assigned a numerical value, and the raw activity sequences from each session were re-coded accordingly.

To better understand the relationship between actual sequential interactions with the applications and outcome measures, we partitioned each dataset into overlapping sub-sequences, creating a library of short activity kernels. We hypothesized that canonical patterns of user interactions could be extracted from this library and used to provide feedback to developers about specific usage patterns of their applications, and for identification of meta-workflows that might be predictive of outcome measures derived from the summer camp experiment (e.g. scores from OT questions, pre- and post-task survey responses) and/or demographic data about the participants (e.g. background, work experience). In the following, we describe the process by which the sub-sequences were extracted, clustering of sub-sequences to identify canonical behavior patterns, and metrics derived from raw and meta-sequences.

We partitioned each dataset into overlapping sub-sequences (SSs) of lengths 3-6 and identified the set of all the unique SSs observed across all users, as well as the number of times each unique sequence occurred. We chose this range of lengths through experimentation with the data. The resulting number of sub-sequences varied widely across tools, as they are a function of the total number of unique activities available to the user.

The set of SSs derived from each session contains a lot of redundancy. Additionally, there are many instances of singleton SSs, i.e. unique SSs that occurred only once across all user sessions. To extract meaningful meta-sub-sequences from this collection, we implemented a novel fuzzy clustering approach that combines methods from natural language processing with community detection in network models [21].

The fuzzy clustering approach we implemented does not technically constitute a 'fuzzy' method in the strictest sense of the word often used in the machine learning literature. It's used to emphasize the fact that one of the goals of the clustering algorithm was to identify a set of *representative* SSs, as opposed to selection of a set of SSs from the collection as the representation of each cluster. We chose to represent the collection of SSs for each application as a network, where each node is one of the unique SSs extracted from all user sessions, and the edges between nodes represent the similarity between the two SSs.

We experimented with several similarity metrics often used for approximate string matching [22] and selected two metrics for use in the clustering process: The length of the longest common sub-sequence (LCS) shared by two SSs. The Jaro distance d_j between the two SSs.

Given two strings, s_1 and s_2:

$$d_j = \begin{cases} 0, & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), & otherwise \end{cases} \quad (1)$$

where $m =$ the number of matching characters, $t =$ the number of transpositions, and $|s_1|$ and $|s_2|$ are the lengths of strings s_1 and s_2. Two characters are considered matching if they are identical and the distance between them is $\leq \left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$. The number of transpositions is the number of matching characters, in different sequence order, divided by 2 [23]. The scale of the Jaro distance metric is between 0 and 1, with 0 being a complete mismatch between sequences, and 1 indicating that the sequences are identical.

Both metrics allow for some degree of mismatch between the two sequences, by design. Since the SSs represent actual sequences of user interactions with an application, slight variations in the sequences (e.g. transpositions, character misalignment) shouldn't preclude them from being placed into the same category. Some sample sequence pairs and their corresponding LCS and Jaro distance values are provided in Table 1 to show how similarity between SSs translates to a numerical value.

Table 1. Sample sub-sequence pairs with corresponding Jaro distance values and longest common sub-sequence.

Sample sub-sequence pairs	Jaro Distance	Longest Common Sub-sequence
[9, 44, 2, 44, 1] [44, 9, 49]	0.52	1
[40, 44, 21, 21, 21] [44, 40, 32]	0.52	1
[20, 21, 48] [26, 48, 21, 9, 45]	0.52	1
[21, 9, 49, 48] [45, 9, 49, 48, 9]	0.78	3
[11, 8, 40, 32, 9] [11, 9, 40, 32]	0.78	3
[8, 44, 40, 32, 44] [44, 9, 40, 44]	0.78	3
[2, 3, 5, 53, 1] [3, 5, 53, 1]	0.93	4
[8, 21, 20, 49] [8, 21, 21, 20, 49]	0.93	4
[2, 3, 2, 44] [5, 2, 3, 2, 44]	0.93	4

Running the community detection clustering algorithm on the network model derived from the full set of unique nodes using either distance metric was relatively unsuccessful. The network was not inherently modular due to large

average closeness between nodes of the network structure, even when strict minimum similarity metric thresholds were used to reduce the number of edges between nodes.

Our solution to this problem was to reduce the number of SSs used to construct the network model, prior to running the community detection algorithm. These 'seed' nodes were selected from the set of length-3 sequences with frequency of occurrence greater than a minimum threshold (chosen to be 4). A small network was constructed from these sequences using LCS length as the distance metric between nodes, and a minimum LCS length = 2 for an edge to be placed between two nodes. The community detection algorithm resulted in a partition that was used as the seed for determining the community membership of the remaining SSs in the collection. Longer sequences (lengths 4-6) that were direct descendants of the length-3 seeds (i.e. have one of the seeds as their first 3 values) with frequencies of occurrence greater than 4 were directly connected to the seed nodes, creating a hierarchical skeleton structure for each community, against which all other SSs were compared.

The remaining unique sequences, the vast majority of which were singletons, were compared to the nodes in the skeleton of each community, using the Jaro distance as the measure of similarity. If the Jaro distance was ≥ 0.8 for a single community, this SS was considered an associate of the community. Figure 28 depicts the network representation of the unique sequences collected from the Neon sessions. Nodes and edges are colored by community membership, and there were 9 total communities identified from this set of sub-sequences. Skeleton nodes and edges are shown in white. The number of SSs assigned to each community varies considerably.

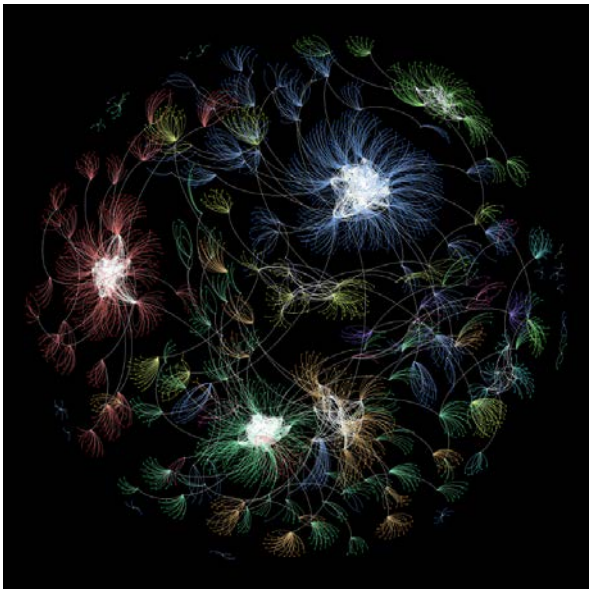


Figure 28: Network representation of the sub-sequences extracted from all of the Neon user sessions.

By constructing the network in this way, we were able to connect the vast majority of the remaining unique sequences with a single community. i.e., there were very few instances of SSs being equally similar to a member of the skeleton of more than one community. The number of communities identified for each application, as well as the proportion of all the unique sequences associated with one of the communities is shown in Table 2.

Table 2. Sub-sequence statistics for all applications.

Application	Number of activity codes	Total number of unique sub-sequences collected from all users/sessions of all lengths (3-6)	Number of communities (clusters)	% of sub-sequences accounted for by the clusters
Neon	60	6558	9	86.1%
Newman	28	3419	6	79.2%
Minerva	127	10840	11	70.8%
Resonant	25	3036	6	82.9%
Aperture Tiles	114	7530	8	86.2%
FEAT	34	5347	7	96.0%

We generated several metrics from the unique set of SSs collected for each tool for each user's session. A description of each of the metrics is provided in Table 3 below. Overall, the metrics are designed to assess the diversity of behavior patterns exhibited by each user during their sessions.

Table 3. Description of metrics derived from sub-sequence statistics.

Unique sub-sequences and singletons	The total number of unique SSs observed during a user's session was determined, as well as the number of times each SS occurred. Singleton activities occurred only once during the session.
prop_unique_L(3, 4, 5, 6, all_L)	number of unique SSs in a session / total number of SSs in the session, and the lengths of those SSs ("L")
prop_singletons_L(3, 4, 5, 6, all_L)	number of SSs occurring 1 time in a session / total number of SSs in the session, and the lengths of those SSs ("L")
Uniqueness of the sub-sequences	Each sub-sequence represents a series of user interactions with the application. A unique-activity (UA) SS is one where there are no repeated activities in the SS, e.g. [1, 5, 11, 2] Similarly, a unique-elementGroup (UEG) SS is one where each activity in the sequence corresponds to an interaction with a different elementGroup, as defined by USER-ALE logging scheme.
prop_num_all_unique_codes_L(3, 4, 5, 6, all_L)	number of all UA SSs / total number of unique SSs in the session, and the lengths of those SSs ("L")
prop_counts_all_unique_codes_L(3, 4, 5, 6, all_L)	total number of observed instances of the UA SSs / total number of SSs in the session, and the lengths of those SSs ("L")
prop_num_all_unique_eg_L(3, 4, 5, 6, all_L)	number of all unique UEG SSs / total number of unique SSs in the session, and the lengths of those SSs ("L")
props_counts_all_unique_eg_L(3, 4, 5, 6, all_L)	total number of observed instances of the UEG SSs / total number of SSs in the session, and the lengths of those SSs ("L")
Bouncing sub-sequences	A sub-sequence is classified as 'bouncing' if the activities in the sequence bounce back and forth between two different activities, e.g. [1, 2, 1, 2]

prop_bounces_L(3, 4, 5, 6, all_L)	total number of observed instances of bouncing SSs / total number of SSs in session, and the lengths of those SSs (“L”)
prop_unique_bounces_L(3, 4, 5, 6, all_L)	number of SSs defined as bouncing / total number of unique SSs in the session, and the lengths of those SSs (“L”)

Part of the exercise of validating new metrics is examining what information those metrics contain. Another part is in understanding whether information between metrics is redundant. In evaluating the first question for sub-sequence metrics, we find that the length of sub-sequences is not a key indicator of user experience or task performance (as we found last year): the relationships between SS length metrics did not scale with either performance variables (e.g., task accuracy, time-to-complete) or experience variables (e.g., cognitive load, engagement). Rather we found that metrics describing specific usage of SSs were related to (or proxies for) performance and user experience variables. Particularly, the composition of sub-sequences users generated and the number of times they used these sub-sequences (i.e., singletons) were predictors of performance and user experience. The relationships between these metrics and performance/experience metrics were consistent regardless of the length of SSs observed. It’s likely that these effects were masked last year based on the way we extracted SSs last year.

We find a few interesting patterns: first, the uniqueness of SS users generated (Uniqueness of Sub-sequences) describes how varied the composition of their SSs were—whether SSs were composed with a variety of activities (e.g., hover, toggle, scroll). We find that users who generated a large proportion of SSs composed of unique activities (different from one another) were more likely to show high cognitive load ($r = .41$, $p < .05$) and report low task engagement ($r = -.38$, $p < .05$). The second key pattern we find is that users that generated a high proportion of SSs composed of unique UI element interactions (e.g., maps, plots, tables) were also more likely to report high cognitive load, specifically frustration with the interface ($r = .51$, $p < .01$). The third pattern we find is that users with a large portion of single use SS (singletons) are more likely to report low cognitive load ($r = -.38$, $p < .05$) and high engagement ($r = .74$, $p < .001$). Users with a higher proportion of single use SS were also more likely to take less time to complete tasks ($r = -.60$, $p < .001$). These findings posit a juxtaposition between the uniqueness or variety within sequences and the number of times individual workflows (e.g., SS) are made use of.

Our interpretation of these findings is that SS metrics for uniqueness capture floundering or frustration. Users that are composing workflows (e.g., SS) of wholly unique elements may be randomly testing patterns of interactions rather than demonstrating systematic, integrated usage of applications. While one might at first expect that using these kinds of workflows would result in a greater distribution of effort across all possible activities (e.g., diffuse or not-peaked use), this doesn’t appear to be the case. In fact, SS uniqueness metrics are

uncorrelated with BP-HMM metrics (r = -.22, p = .21). The difference between the two is likely explainable based on the fact that BP-HMM metrics account for how activities are integrated in time, not just raw frequencies.
SS singleton metrics likely capture an appropriate fit of UI workflow to tasks. Singleton metrics are inversely correlated with BP-HMM metrics (r = -.41, p < .05) and somewhat redundant with BP-HMM metrics for peakedness, suggesting that they tap into how appropriately activities were selected and used together in time.

Using step-wise regression, we also observe that BP-HMM peakedness metrics and SS uniqueness metrics are completely independent measures of cognitive load. Thus, when added together in multiple regression equations for cognitive load, they collectively account for nearly 40% of the variation in self-reported cognitive load. Based on our year 2 findings (36% of variance explained), this is 4% better prediction than pupillometry (e.g., eye-tracking) on the same measure.

4.2.3. XDATA Application Evaluation. Our analytic workflow for evaluation is to first provide descriptive statistics about applications across metrics (performance, difficulty, usability, etc.) that are easiest to interpret because their information is intrinsic to the measure, e.g., questionnaires, performance metrics. Following that we report descriptive statistics for metrics that emerge from intuitive models, however, they are not explicitly labeled by users. We then take steps to normalize data prior to determining ranks for applications, based on their performance against metrics. Finally, we provide rankings for which applications performed best and worst.

In year 3, 6 applications were tested with users recruited by DARPA. These applications were developed to address one of three challenge problems, issued by DARPA—Population Movements, De-Aliasing, and Financial Fraud. Evaluations were performed with care so as to invite fair comparisons between applications within and between challenge problems. In year 4, the same applications were tested on a massive scale using an MTURK sample of over 1000 persons. Below we present findings and comparisons between applications on a variety of metrics computed from measures described above (see 4.2.2). Findings are presented for each evaluation activity, side-by-side. Overall, we observed few differences in major findings between the two activities. However, where differences do occur, we defer to findings from year 4 for which there was substantially more statistical power due to the larger finding. Additionally, it is worth noting that with the large samples, standard measures of statistical significance are generally inflated. As such, we report effect sizes, which provide a clearer depiction of real differences.

Differences between applications are presented using box plots. Box Plots provide a better characterization of differences between classes as they depict the distribution of scores for each class (e.g., applications) and central tendency measures that make skewed distributions easier to identify than histograms or bar charts. Whiskers on box plots provide detail on the interquartile range (Tukey Boxplot), e.g., the spread of the distribution of scores on each metric for each application. The “box” itself represents the true central tendency or mass of the

distribution. The solid line represents the *median* score of the distribution, which is a measure of central tendency that is less sensitive to skews than means, or averages. Hollow dots represent outliers in the tails. Where skews exist, we did not attempt to normalize distributions for evaluation purposes because the goal of evaluation is accurate description, not inference (e.g., prediction). Outliers depicted in box plots were not removed, either, because data used for evaluation was filtered for cases reflecting non-compliance. Examples of non-compliance are participants not spending sufficient time on tasks (<5mins of 30mins total), or participants simply "clicking through" tasks, as measured by >60 clicks per minute average mouse-click activity rate, independent of other activities (e.g., pan, scroll, etc.). Our report below reflects the most accurate accounting of findings.

In year 3 and 4 we first analyzed task performance metrics and users' subjective ratings of their experiences with the application. As they are concretely labeled measures of usability, they provide context for interpreting findings related to activity during tasks.

We began each evaluation analysis by examining task performance, as measured by the number of correctly answered questions within operational tasks given to participants, of which there were two tasks each composed of 5 questions (10 questions total). Our experimental design has a repeated measures component: users perform two tasks in sequence. This allows us to examine whether proficiency with applications and performance improves with practice. The order with which these tasks are presented to users are counterbalanced across users so as to rule out whether the order of tasks has a systematic impact on the performance of tasks. In Year 3, we found significant differences between applications ($F(5,34) = 8.30$, $p < .000$): users of Resonant and Newman performed best (Figure *29*). Corrected, post-hoc comparisons clearly indicate that Newman users were statistically more accurate than all other applications, save Resonant and Neon. Resonant users were more accurate than Aperture users, as were Neon users. In Year 4, we found a sizeable effect for statistically meaningful differences between applications ($F(5,653) = 106.8$, $p < .000$). Again, post-hoc tests confirm users of Resonant and Newman performed best, and there were no differences between other applications (Figure *29*).

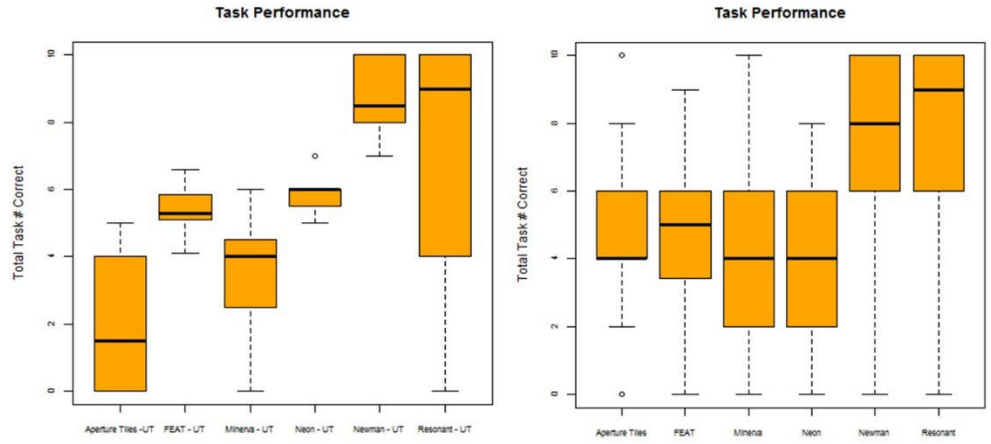


Figure 29. Task Performance Differences Between Applications (Yr 3-Left, Yr 4-Right).

Next, we examined whether participants' performance improved between repeated exposures with the applications. In year 3, a pair-samples t-test across each of users' sessions revealed no differences in performance between sessions (t (39) = .24, p = .81). In year 4, we discovered very small differences in performance between sessions (given the sample size; t (658) = 4.24, p < .000). These findings suggest that participants did not improve in task performance across *different* tasks, even with repeated exposure with the same application. We then examined whether session-related performance differences were systematically different between applications.

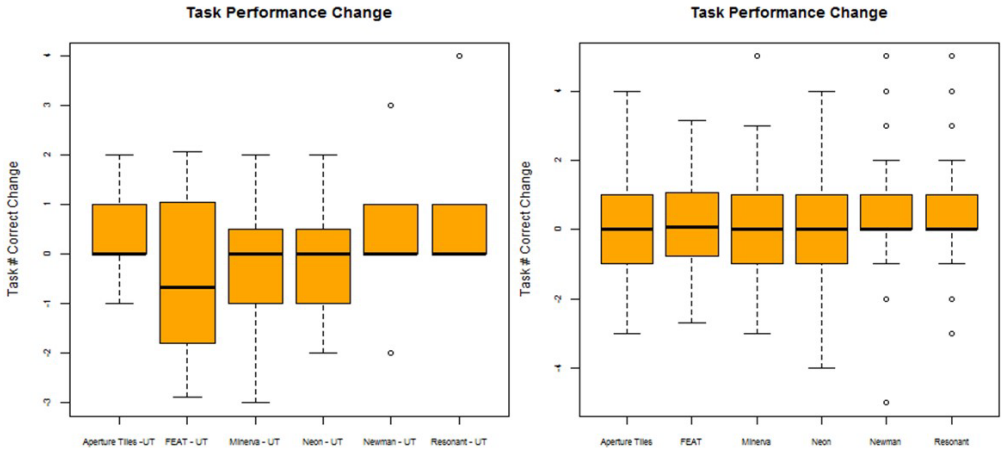


Figure 30. Task Performance Differences Between Sessions, by Application (Yr 3-Left, Yr 4-Right).

In year 3, we found no differences in task-related improvement that would be attributable to differences across applications (F (5,34) = .72, p = .62); post-hoc pair-wise analyses confirm this is the case. We found the same or similar pattern in year 4. Though we detected trivial differences (F(5,653) = 4.44, p < .000) between applications, post-hoc comparisons reveal no statistically meaningful differences between pairs of applications (Figure 30).

Next, we analyzed how much time participants spent, on average completing tasks, which is relevant for understanding participants' efficiency while using specific applications. Time-on-Task was computed by extracting time stamps from the start and end times of each of 10 task components that were completed by users (or started), finding the difference in time between those times, and summing across all 10 task component time differences. In Year 3, we find no statistical differences on this comparison (F(5,34) = 1.31, p = .28) in the ANOVA model (Figure *31*). However, corrected post-hoc comparisons confirm that Feat users were far more likely to take more time in completing their tasks than all applications except Minerva. We found no other statistical differences between applications in year 3. In year 4, we find small, statistically meaningful differences (F(5, 658) = 37.8, p < .001). Post-hoc pair-wise comparisons highlight differences between de-aliasing applications (Newman, Resonant) and other applications, such that participants using these applications spent less time-on-task. There were also small differences between FEAT, which required the most time-on-task and population movements applications (Aperture, Minerva, and Neon) *(*Figure *31*).

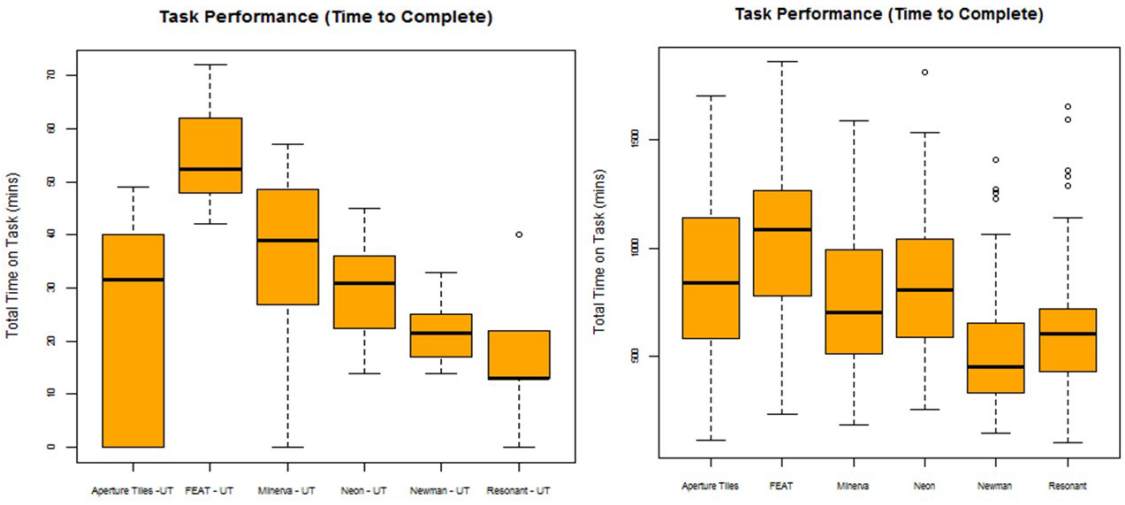


Figure 31. Time-to-Complete Task Differences, by Application (Yr 3-Left, Yr 4-Right).

The next analyses we perform in course of evaluation is to investigate how participants rated the application, and label their experiences with applications through the various post-task questionnaires they complete. Not only are these analyses descriptive, but they also provide valuable context for interpreting

findings across measures, given their collinearity. This is also for normalization in rankings.

The first of these analyses is to examine participants' self-reports of task-related difficulty; they were asked to rate how difficult they felt their tasks were, independent of the application they were asked to complete the task with. In year 3, our small sample of analysts reported statistically meaningful differences between applications ($F(5,19) = 5.83$, $p < .01$). Post-hoc pairwise comparisons indicated that differences were primarily driven by dealiasing tasks associated with the Newman and Resonant applications, which were rated as less difficult than most other applications, particularly Aperture, Minerva, and FEAT. In year 4, our large MTURK sample also reported statistically meaningful differences in the difficulty of tasks associated with challenge problems ($F(5, 591) = 93.6$, $p < .000$). Post-hoc pairwise comparisons indicate that these differences were largely driven by dealiasing tasks (Newman, Resonant applications), which were rated much less difficult than other tasks. However, we found that population movement tasks with Neon were rated significantly less difficult than the same tasks paired with Aperture, and financial fraud tasks paired with FEAT.

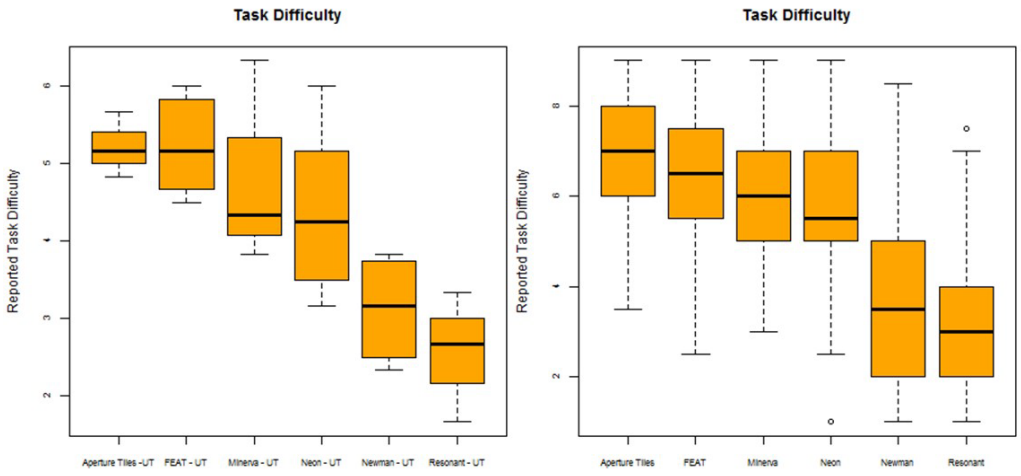


Figure 32. Differences in Self-Reported Task Difficulty, by Application (Yr 3-Left, Yr 4-Right).

These findings suggest that certain tasks were systematically easier than others, with dealiasing tasks rated least difficult. Also, while tasks were almost identical across tools situated within specific challenge problems (e.g., population movements), participants systematically rated tasks paired with specific applications (Aperture) as being difficult, while other participants performing the same tasks with different applications as less difficult. Again, participants were asked to report on how difficult the task was, independent of the application they were asked to perform it with. *This suggests that participants were unable to de-couple the intrinsic difficulty of the task with the difficulty imposed on the task by the application.* We do not believe that these effects are driven by expert- or

novice-effects, given that similar within class (e.g., challenge problem) differences in difficulty were observed for both our year 3 sample—with self-selecting analysts (e.g., experts), and our year 4 sample with public MTURK users. Generally, our findings from year 4 (MTURK users) replicate our findings from year 3 (Figure 32).

We next investigated participants' self-reported cognitive load, which is the added task difficulty or mental effort imposed by the application used to complete tasks, independent of how intrinsically difficult the task itself is. We collected labeled user data about their cognitive load through post-task questionnaires, following their experiences with applications. Participants were asked to rate each operational task component based on facets of cognitive load. In year 3, our small sample of analysts reported statistically meaningful differences in how the various XDATA applications introduced cognitive load into their tasks ($F(5,26) = 4.76$, $p < .01$). Corrected post-hoc comparisons indicate that Newman users were substantially less likely to indicate that their tasks required additional mental effort than all other applications except Neon and Resonant. In year 4, our large sample of MTURK participants also reported statistically meaningful differences in how XDATA applications introduced cognitive load into their tasks ($F(5, 658) = 41.31$, $p < .001$). Again, corrected post-hoc comparisons indicated that users of Newman and Resonant reported the least cognitive load. However, with a larger sample size, our year 4 evaluation indicates that users of Aperture reported the most cognitive load. Other differences between applications within class (challenge problems) were not significant (Figure 33).

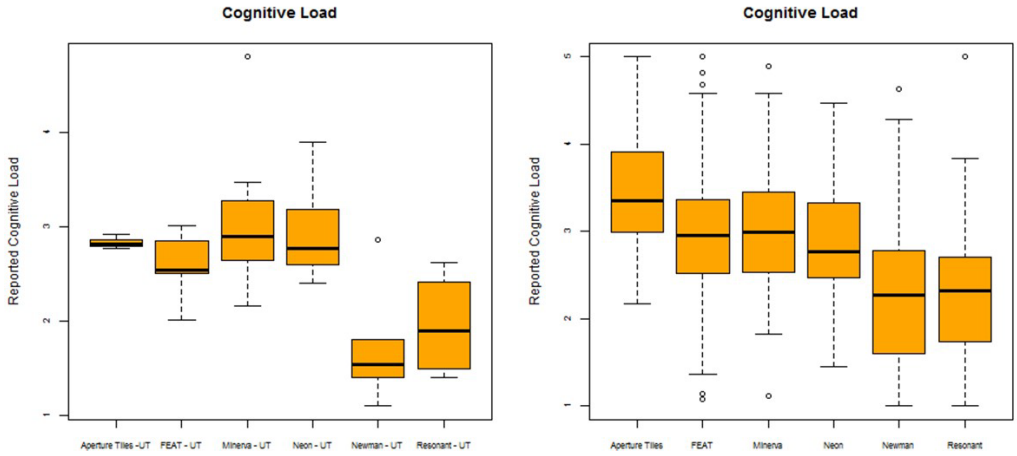


Figure 33. Differences in Self-Reported Cognitive Load, by Application (Yr 3-Left, Yr 4-Right).

In our next examination we explored participants' reports of task engagement, which reflects the degree to which participants were immersed within their tasks and lose a sense of time. In year 3, our ANOVA models indicate significant differences between applications on engagement measures ($F(5,27) = 3.99$, $p < .01$). Corrected, post-hoc comparisons indicate that differences between Aperture and Minerva, and Newman and Minerva are driving these effects. Users

of Aperture were most consistently engaged overall. In year 4, our large sample of MTURK participants, clarified differences between applications. Again, statistically meaningful differences (although small, given the sample size) between applications were observed ($F(5, 658) = 12.23$, $p < .001$). Corrected post-hoc comparisons indicate that participants who worked with de-aliasing applications (Minerva and Resonant) reported the most engagement. Corrected post-hoc comparisons suggest that participants who worked with de-aliasing applications reported more engagement than those who worked with Aperture, FEAT, and Minerva, but not Neon (Figure 34).

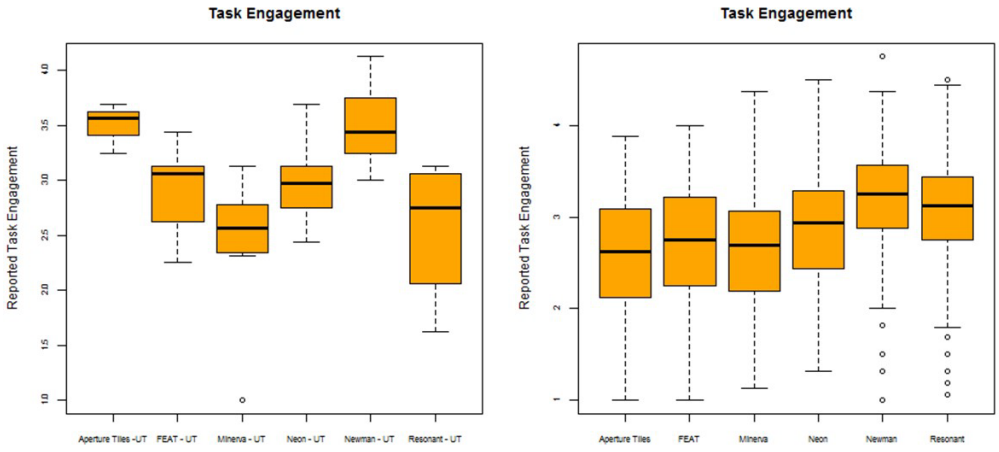


Figure 34. Differences in Self-Reported Task Engagement, by Application (Yr 3-Left, Yr 4-Right).

We next examined self-reported differences in task enjoyment—whether participants simply enjoyed their tasks (with their applications). In year 3, we did not observe any meaningful differences between applications with respect to self-reported enjoyment ($F(2,27) = .806$, $p = .5$). However, in year 4, with our large MTURK sample we found small differences between applications with respect to task enjoyment ($F(5,590) = 9.38$, $p < .000$). Overall, participants reported that they enjoyed tasks with dealiasing applications (Newman and Resonant) and Neon the most. These three applications were enjoyed significantly more than Aperture, and Newman was enjoyed significantly more than both Minerva and FEAT applications (Figure *35*).

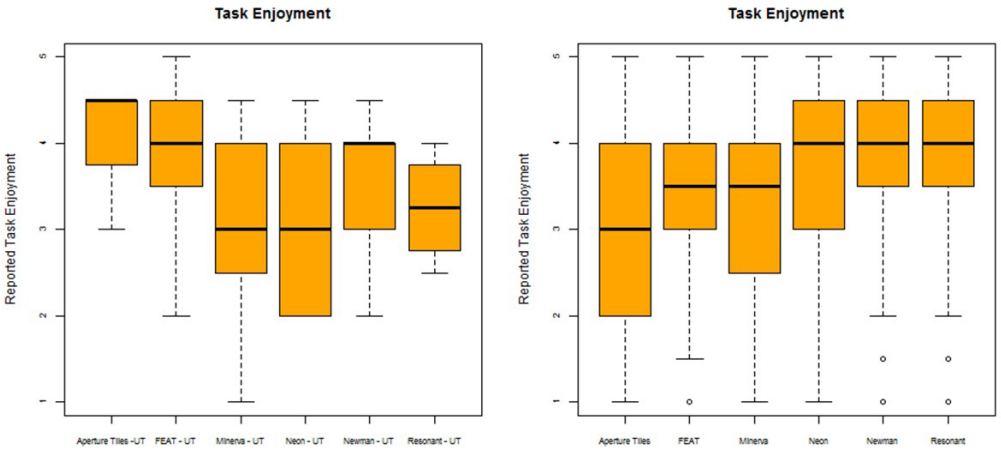


Figure 35. Differences in Self-Reported Task Enjoyment, by Application (Yr 3-Left, Yr 4-Right).

Finally, we evaluated the XDATA applications with an objective metric of usability, using data collecting with Apache SensSoft technology. We utilized a well-validated metric of integrated (or dis-integrated) use stemming from our BP-HMM modeling technique, describing how participants integrate user interface elements and allocate their effort across these elements, in time. In this case, the metric is signed to describe when users are *not* integrating various UI elements (e.g., low scores are better), as they might if they understood how the various UI elements were meant to be used together. This metric shows a strong, positive association with subjective measures of workload and cognitive load, and a negative association task performance, and outperforms simple “click-counting” metrics of activity rate in predicting subjective user reports.

In both year 3 and year 4, we found this metric to be most sensitive to differences between applications, compared to other measures (as evidenced by large F-statistic effect sizes) (Figure *36*). While more sensitive, this metric is both correlated with self-report measures of usability, and generally illustrate similar trends regarding which applications are most usable. In year 3, large, statistically meaningful differences on were observed in tests comparing all applications ($F(5,30) = 41.56$, $p < .000$). Post-hoc comparisons revealed that Aperture users evidenced the least integrated use, but was statistically indistinguishable from FEAT. These two applications were used in the least integrated way. In year 4, with a larger sample, we observed similar, but much larger differences across applications, e.g., stronger effect sizes ($F(5, 667) = 205.2$, $p < .000$). This is notable, given that effects tend to be smaller with larger samples sizes. Corrected post-hoc comparisons reveal differences between all pairs of applications, except for Aperture and Minerva, which were not different from one another in terms of integrated use. FEAT users evidenced the most dis-integrated use, while Resonant users evidenced the most *integrated* use.

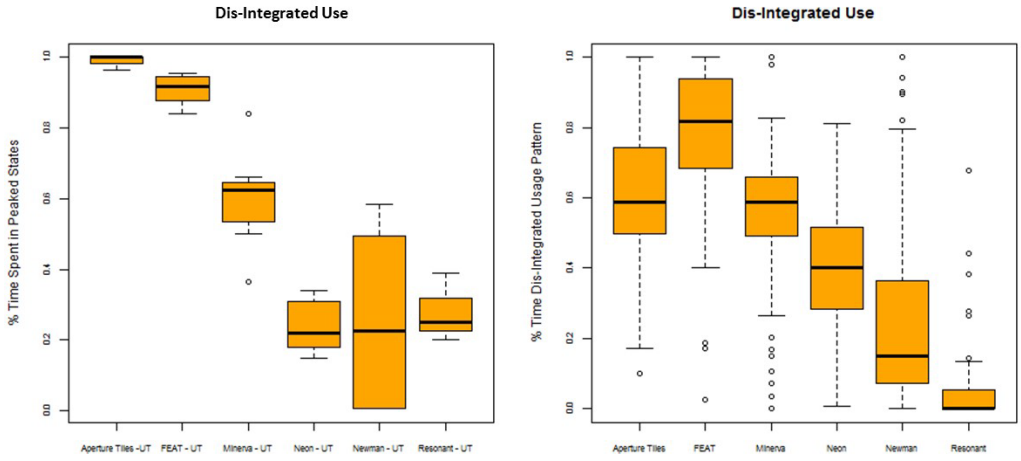


Figure 36. Differences in Objective Integrated Use, by Application (Yr 3-Left, Yr 4-Right).

The final step we take in evaluating applications is to rank applications based on usability. These ranks are meant to provide program personnel with information about the best applications overall—best fit to the tasks users were provided with them—as well as the best applications within class (e.g., challenge problem). In order to account for the most information in summarizing ranks among applications, and to provide a fair assessment of applications, we apply various weights within the ranking process. Generally, ranks are calculated by ranking applications across each metric we calculate for each application (performance, user-experience (subjective ratings), BP-HMM Metrics), then we average the ranks for each metric across each application and rank against those averages to provide summary ranks for each application (Net Ranks). This provides a summary of rankings across metrics, while preserving the scaling and signage of each metric (whether high scores indicate positive or negative outcomes). However, because ordinal rankings do not take into account the actual distance between ranked items (how much they differ from one another numerically), subtle and trivial differences between applications can cause rankings that are potentially misleading. As such, two weights are applied to raw metric values for each application prior to ranking applications against each metric.

The first weight addresses the sensitivity of the metric—the ability for the metric to discriminate between applications given its scaling. In this respect we weight each metric value by the metrics' corresponding F-statistic (see above sections) for between application differences. Thus in aggregate, this weighting scheme weights the input to averages based on metrics with the most discriminant power. The second weight that is applied is for task difficulty. These weights are meant to reflect the intrinsic difficulty of the tasks that applications are paired with owing to the data and challenge problem they are developed for. In certain cases, some applications are meant to address less complex tasks (e.g., dealiasing vs. populations) and it is a concern as to whether certain

advantages are given to these applications in rankings. Thus, task difficulty ratings are averaged across all ratings taken from the same challenge problems, as these were similar tasks. These averages are then multiplied by metric values that are positively signed (e.g., more weight is given to task performance metrics for tasks that were more difficult), and are divided from metrics that are negatively signed (e.g., less weight is applied to cognitive load scores for applications paired with more difficult tasks).

In summary, in year 3 application rankings are given in Figure 37. The bounded cells (in orange) reflect metric values for key usability metrics (cognitive load, dis-integrated use). The Net Rank column indicates rankings for each application across metrics, without weighting. Weighted Rank columns indicate ranks after accounting for metric sensitivity (F) and both metric sensitivity and task difficulty (diff). Overall, in year 3, de-aliasing applications tended to perform better against other applications across different weighting schemes, with Newman performing best in that class. FEAT consistently underperformed relative to other applications. Within the Population Movements challenge problem, Neon consistently performed better than either Aperture or Minerva.

Challenge Problem	Task Difficulty	Application Name	Task Perform	Time on Task (min)	Engagement	Enjoyment	Cognitive Load	Dis-integrated Use	Net Rank	Weighted Rank (F)	Weighted Rank (diff)
Pop Mov	5.22	aperture tiles	2.00	25.33	3.50	4.00	3.93	0.99	4	4	4
Pop Mov	5.50	minerva	3.50	35.75	2.42	3.07	3.35	0.62	5	5	3
Pop Mov	4.39	neon	5.86	29.57	2.99	3.08	3.24	0.25	2	3	1
Dealias	3.08	newman media	8.67	22.00	3.50	3.58	1.60	0.26	1	1	2
Dealias	2.29	resonant	6.60	17.60	2.56	2.27	2.10	0.28	3	2	3
Financial	5.40	feat	5.40	54.88	2.89	2.61	3.52	0.91	6	6	5

Figure 37. Year 3 Application Rankings Against Key Usability Metrics

Year 4 application rankings are given in Figure 38. Again, the bounded cells (in orange) reflect metric values for key usability metrics (cognitive load, dis-integrated use). The Net Rank column indicates rankings for each application across metrics, without weighting. Weighted Rank columns indicate ranks after accounting for metric sensitivity (F) and both metric sensitivity and task difficulty (diff). Overall, in year 4, de-aliasing applications tended to perform better against other applications across different weighting schemes, although not as strongly when weights were applied reflecting task difficulty (these were with Newman performing best in that class). Aperture consistently underperformed relative to other applications across weighting schemes. Within the Population Movements challenge problem, Neon consistently performed better than either Aperture or Minerva.

Challenge Problem	Task Difficulty	Application Name	Task Peform	Time on Task	Engagement	Enjoyment	Cognitive Load	disintegrated Use	Net Rank	Weighted Rank (F)	Weighted Rank (diff)
Pop Mov	6.94	Aperture Tiles	4.46	881.00	2.58	3.10	3.41	0.60	6	6	6
Pop Mov	6.12	Minerva	4.58	773.00	2.62	3.40	3.01	0.56	5	4	3
Pop Mov	5.65	Neon	4.32	844.00	2.86	3.65	2.86	0.41	3	3	2
Dealias	3.69	Newman Media	7.87	510.00	3.16	3.90	2.24	0.26	1	1	4
Dealias	3.38	Resonant	7.46	625.00	3.05	3.79	2.28	0.04	2	2	5
Financial	6.37	Feat	4.82	1044.00	2.70	3.29	3.01	0.78	4	5	1

Figure 38. Year 4 Application Rankings Against Key Usability Metrics

Overall, it is Draper's determination that the Dealiasing applications performed better than other applications, with Newman being the best application. The two applications that consistently underperformed are Aperture and FEAT. Within the Population Movements class of application, Neon consistently outperformed both Aperture and Minerva. This determination disregards weighting applied to correct for differences owing to task difficulty. The reason for this is that it is abundantly clear (especially after year 4 analyses) that subjective ratings for applications' usability where highly collinear with ratings for difficulty, regardless of how questions were phrased. In particular, participants' ratings of cognitive load were meant to be anchored in how difficult they though the *application* made the task, regardless of the intrinsic difficulty of the task. In contrast, task difficulty ratings were solicited with juxtaposing prompts—how difficult the task is regardless of the tool used to perform it. Given various statistical analyses reported above, it was the case that participants were unable to disentangle the two—this is obvious given the metric values for task difficulty relative to task performance, cognitive load, etc., and that these value vary within challenge problems or application class, even though all applications of the same class were paired with nearly identical tasks. The same pattern was also observed in both expert samples (e.g., analysts) and novice samples (e.g., MTURK). As such, weighting based on difficulty is problematic. At this point it is clear that applications that were difficult to use made the task more difficult in general, which is clear evidence that specific applications (e.g., Aperture) introduced cognitive load into tasks. Therefore, our determination of the application rankings are driven by ranks weighted by metric sensitivity (F), not those that also incorporate weights for task difficulty.

4.3. Publicity and Community Development

Draper made significant progress in raising the visibility of SensSoft, as well as exploring communities of users in which the technology would have the most impact. Over the course of the 4th year of the XDATA program, the Apache SensSoft community grew only modestly, however, this is largely due to the fact that the bulk of our capabilities were built and transitioned to ASF within the same year. Efforts to continue the Apache SensSoft project and grow this community will persist well after the completion of the XDATA program.

4.3.1. Inclusion into the Apache Software Foundation (ASF). In May 2016, Draper was invited to submit a proposal to include corpus of SensSoft source code into the ASF. Our proposal was accepted by unanimous vote in September 2016, and Software as a Sensor™ became a part of the ASF as an Apache Incubator project. Our Apache champion for this project is Lewis McGibbney (NASA Jet Propulsion Laboratory (JPL)), and our other mentors are Wayne Burke, Paul Ramirez, and Chris Mattmann. Code was officially transitioned to Apache in October 2016 by Draper and our code base was transferred to Apache infrastructure by November 2016. From December to the submission date of this report, Draper has been preparing for our first official Apache public release of our products, this task includes ensuring licensing, build stability, and updating

documentation. The project can be found at http://incubator.apache.org/projects/senssoft.

4.3.2. Exhibitions, Demonstrations and Public Outreach. As part of our publicity and community development efforts, Draper sought to both establish a presence both online, as well as in specific communities including, Internet of Things, UI and User Experience (UX) practitioners, and business analytics interest groups.

Inclusion in the ASF provided a key vehicle to grow a community of interest and online presence, in part because of its visibility and opportunities to network with the larger ASF open-source community. As a "landing-pad" for our online presence, we established a comprehensive website for posting information about the Apache SensSoft project, including community information, software documentation, links to software repositories (git, github), and interactive demonstrations (Figure 39, http://senssoft.incubator.apache.org). Given the nature of our technology, our demo allows visitors of the page to interact with the page, see their behavior through a live logging server, and a means to visualize the behavior of page visitors through the "Bowie" plot (Figure 40).

Figure 39. Screenshot of the Apache SensSoft Webpage

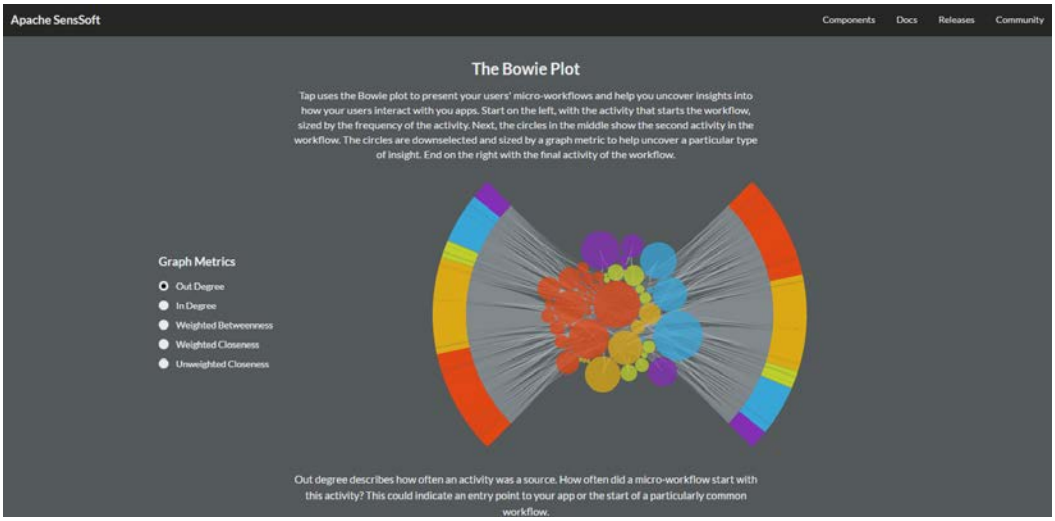


Figure 40. Interactive Demos Hosted on the Apache SensSoft Webpage

We also formed a social media presence through our Apache message boards, a Twitter account (@ApacheSensSoft), as well as a YouTube channel (Apache SensSoft) for posting videos and demos (see http://senssoft.incubator.apache.org/community/). We coordinated these investments with a number of exhibitions at trade-shows throughout 2016 for interest groups related to the Internet of Things (Sensors Expo; San Jose, CA, 5/2016), business analytics (eMetrics Summit; New York, NY, 10/16), and UI/UX (UIE21; Boston, MA, 11/16). For these events, Draper prepared distinctive branding and marketing materials (Figure 41) to generate interest in the Apache SensSoft project, as well as booth set up for exhibitions (Figure 42).

Figure 41. Apache SensSoft Marketing Materials

Figure 42. SensSoft Exhibition Booth Display

4.3.3. Transition Opportunities. In addition to public outreach, Draper also reached out to transition partners within the USG and DoD. We sought to provide a means for providing the innovations and distinctive capabilities developed as part of the XDATA program to agencies that would benefit from these efforts. In this regard we were successful. In the fall of 2016, NGA Research included the SensSoft project as an added task item in an existing collaborative research and development agreement (CRADA) held between NGA and Draper. The scope of this task involves providing business analytics and user testing services to NGA for their current applications, as well as research and development activities to benefit future capabilities utilizing SensSoft. In January 2017, NGA approved SensSoft software for use within the agency. We are currently working to deploy SensSoft within AWS resources provisioned by NGA. Draper has also received additional funding through both DARPA (RSPACE) and IARPA (MOSIAC) for SensSoft projects.

5. CONCLUSIONS

Over the course of four years of performance on DARPA's XDATA program, Draper consistently focused our technical effort to meet the challenges posed by Dr. Chris White and Dr. Wade Shen. Blending rigorous scientific research and robust software development, Draper was able to innovate new, non-invasive methods for evaluating the usability of software applications. Draper's Software as a Sensor™ technology provides the means for collecting high-granularity software activity logs as well as a back-end, scalable infrastructure that makes these methods available to a wide open-source community through the Apache SensSoft project. Modeling approaches we developed to derive insights from software activity logs show comparable or discriminating abilities to predict key usability metrics, as well as advantages over canonical laboratory and self-report methods. In this respect, Draper has substantively contributed a major advancement in the field of human computer interaction. With viable transition opportunities in the commercial, DoD, and open-source communities, Draper will continue to develop this capability and push forward the fields of human computer interaction and human system integration.

6. References

[1] J. T. Cacioppo*, et al.*, "The efficient assessment of need for cognition," *Journal of Personality Assessment,* vol. 48, pp. 306-307., 1984.

[2] S. Epstein*, et al.*, "Individual differences in intuitive-experiential and analytical-rational thinking styles.," *Journal of Personality and Social Psychology,* vol. 71, pp. 390-405., 1996.

[3] S. Frederick, "Cognitive reflection and decision making," *Journal of Economic Perspectives,* vol. 19, pp. 25-42, 2005.

[4] G. Nenkov*, et al.*, "A short form of the maximization scales: Factor structure, reliability and validity studies.," *Judgment and Decision Making,* vol. 3, pp. 371-388, 2008.

[5] P. Norris*, et al.*, *The rational-experiential inventory, short form. Unpublished Inventory.* Amherst, MA.: University of Massachusetts at Amherst., 1998.

[6] A. Roets and A. Van Hiel, "Item selection and validation of a brief, 15-item version of the need for closure scale.," *Personality and Individual Differences,* vol. 50, pp. 90-94, 2011.

[7] D. Webster and A. Kruglanski, "Individual differences in need for cognitive closure," *Journal of Personality and Social Psychology,* vol. 67, 1994.

[8] A. Fagerlin*, et al.*, "Measuring numeracy without a math test: Development of the subjective numeracy scale (SNS). ," *Medical Decision Making,* vol. 27, pp. 672-680., 2007.

[9] B. J. Zikmund-Fisher*, et al.*, "Validation of the subjective numeracy scale (SNS): Effects of low numeracy on comprehension of risk communications and utility elicitations.," *Medical Decision Making,* vol. 27, pp. 663-671., 2007.

[10] A. H. Roscoe and G. A. Ellis, "A subjective rating scale for assessing pilot workload in flight: A decade of practical use," DTIC Document1990.

[11] J. H. Brockmyer*, et al.*, "The development of the Game Engagement Questionnaire: A measure of engagement in video game-playing," *Journal of Experimental Social Psychology,* vol. 45, pp. 624-634, 2009.

[12] J. Nielson and R. Molich, "Heuristic evaluation of user interfaces.," in *ACM CHI*, Seattle, WA, 1990, pp. 249-256.

[13] L. J. Mariano*, et al.*, "Modeling Strategic Use of Human Computer Interfaces with Novel Hidden Markov Models," *Frontiers in psychology,* vol. 6, 2015.

[14] L. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *ASSP Magazine, IEEE,* vol. 3, pp. 4-16, 1986.

[15] R. Kelley*, et al.*, "Understanding human intentions via hidden markov models in autonomous mobile robots," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, 2008, pp. 367-374.

[16] K. Bernardin*, et al.*, "A sensor fusion approach for recognizing continuous human grasping sequences using hidden Markov models," *Robotics, IEEE Transactions on,* vol. 21, pp. 47-57, 2005.

[17] J. Schlenzig*, et al.*, "Recursive identification of gesture inputs using hidden markov models," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, 1994, pp. 187-194.

[18] S. Schliehe-Diecks*, et al.*, "On the application of mixed hidden Markov models to multiple behavioural time series," *Interface focus,* p. rsfs20110077, 2012.

[19] M. C. Hughes*, et al.*, "Effective split-merge Monte Carlo methods for nonparametric models of sequential data," in *Advances in Neural Information Processing Systems*, 2012, pp. 1295-1303.
[20] K. T. Durkee*, et al.*, "System decision framework for augmenting human performance using real-time workload classifiers," in *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision*, 2015, pp. 8-13.
[21] V. D. Blondel*, et al.*, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment,* vol. 2008, p. P10008, 2008.
[22] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR),* vol. 33, pp. 31-88, 2001.
[23] W. Cohen*, et al.*, "A comparison of string metrics for matching names and records," in *Kdd workshop on data cleaning and object consolidation*, 2003, pp. 73-78.

7. APPENDIX A – Publications and Presentations

Poore, J.C., Bowers. C. (2016) Editorial: Virtual Environments as Study Platforms for Realistic Human Behavior. *Frontiers in Psychology,* 7, DOI: 10.3389/fpsyg.2016.01361, URL: *https://www.researchgate.net/publication/308660420_Virtual_Environments_as_Study_Platforms_for_Realistic_Human_Behavior*

Poore, J.C., Bowers, C. (Editors). Virtual Environments as Study Platforms for Realistic Human Behavior. Frontiers in Psychology Research Topic. URL: *https://www.researchgate.net/publication/308660420_Virtual_Environments_as_Study_Platforms_for_Realistic_Human_Behavior*

Poore, J.C. (2016). Human Signal Data Collection in the Wild. Paper presented at the 2016 Human Factors and Ergonomic Technical Advisory Group Conference, Langley, VA.

Poore, J.C. (2016). Embedding Human-System Integration Metrics in Agile Software Evaluation Environments: The Value of Opportunistic Data Collection. Paper presented at the 2016 National Defense Industrial Association Human Systems Conference, Springfield, VA.

Schwartz, J.L., Poore, J.C., Mariano, L.J. (2015). Evaluating How Analysts Make Use of their Tools to Inform Tool Development, Integration, and Adaptation. Poster presented at the 2015 Science of Multi-Intelligence (SOMI) Workshop, Chantilly, VA.

Poore, J.C., Mariano, L.J., Schwartz, J.L. (2015). Operationally Relevant Metrics for Analyst Test-Beds: Evaluating How Analysts Make Use of their Tools to Inform Tool Development, Integration, and Adaptation. Paper presented at the 2015 National Defense Industrial Association Human Systems Conference, Alexandria, VA.

8. LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

ASF	Apache Software Foundation™
ALv2	Apache License (version 2)
API	Application Program Interface
AWS	Amazon Web Services
BP-HMM	Beta-Process Hidden Markov Model
CRADA	Collaborative Research and Development Agreement
D3	Data Driven Documents
DARPA	Defense Advanced Research Projects Agency
DOM	Document Object Model
F	The F statistic refers to a Type III Wald test, otherwise known as an Analysis of Variance (ANOVA). Accompanying notation in parentheses (e.g., F(x,y)) indicate the number of degrees of freedom available for parameter estimation within groups and between groups (Statistics). *Interpretation*: The F statistic summarizes the net separation between distributions of data organized between different groups of data, each with their own average value and standard deviation. The test is meant to ascertain whether the groups are members of the same set or different sets. A low *F* statistic would indicate less separation between groups, and less confidence that the groups were sampled from different distributions.
HMM	Hidden Markov Model
HSIRB	Human Subjects Independent Review Board
IARPA	Intelligence Advanced Research Projects Agency
IR&D	Internal Research and Development
JPL	Jet Propulsion Laboratory
JSON	Java Script Object Notation
LCS	Longest Common Sub-Sequence
LOE	Level of Effort
MOT	Massive Online Testing
MTURK	Amazon Mechanical Turk
NGA	National Geospatial Intelligence Agency

p	The P value (*p*) is the probability of obtaining an effect at least as extreme as the one observed in a sample data, given that the null hypothesis is true (Statistics). *Interpretation*: A small P value would indicate that if the null hypothesis were true (no statistical effect), then a similarly sized statistical effect would be a rare occurance.
r	The Pearson Correlation Coefficient (r) is the linear effect size between two variables (vectors or array), as indicated by the slope of a regression line describing the relationship between those two variables in standardized Euclidean space (Statistics). *Interpretation*: A correlation approaching 1.00 would indicate perfect prediction such that for every one unit increase, in standard deviation, for one variable, the other variable would increase, in standard deviation, by one unit as well. A negative correlation of -1.00 would indicate that for each unit increase (standard deviation) of one variable, the other would decrease by one unit (standard deviation).
R^2	The squared value of the Pearson Correlation Coefficient (r multiplied by itself) (Statistics). *Interpretation*: results in a percentage that indicates the shared covariation between two correlates, or the percent of variation in the criterion variable attributable to the variation in the predictor variable, per se.
REST	Representational State Transfer
S&T	Science and Technology
SCO+CH	Scale Computation and Codebook Handling
SENSSOFT	Software as a Sensor™
SM	SurveyMonkey.com
SS	Sub-Sequence
STOUT	Subject Tracking and Online User Testing (application)
t	The *t* value indicates the results of Student's t-test for difference between two groups. It is named for the specialized test distribution used to generate its value. It is accompanied by a parenthetical indicating the number of degrees of freedom used in estimating its parameters (Statistics). *Interpretation:* The *t*-test tests for differences between two groups of values. A small *t*-test value would indicate trivial or no differences between groups, and little confidence that the two groups are not sampled from the same set.
TAP	Test Application Portal
UI	User Interface
UserALE	User Analytic Logging Engine (application)
UserALEv3	User Analytic Logging Engine (application; version 3)

UserALE,js	User Analytic Logging Engine (application; Java Script)
USG	United States Government
UX	User Experience